UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Facultat d'Informàtica de Barcelona

Grau en Enginyeria Informàtica

Especialitat en Tecnologies de la Informació

Treball de Fi de Grau

Caracterización de Sistemas de Localización en Android

Por:

Adrià Marcoval Martínez

Tutor:

Jorge García Vidal (DAC)

Codirector:

José M. Barceló Ordinas (DAC)

Q2 2018-2019

Resum

L'objectiu d'aquest projecte es basa en la caracterització de la tecnologia Geofence com a un sistema de localització en Android. Com que no hi ha informació real del comportament d'aquest sistema, es dóna un primer pas a la seva experimentació davant de diferents escenaris, per tal de ser capaços d'entendre com actua Geofence en la pràctica, i a més d'això, implementar un algorisme que sigui capaç de rastrejar una persona amb un telèfon mòbil.

Resumen

El objetivo de este proyecto se basa en la caracterización de la tecnología Geofence como un sistema de localización en Android. Como no hay información real del comportamiento de este sistema, se da un primer paso a su experimentación ante diferentes escenarios, para ser capaces de entender cómo actúa Geofence en la práctica, y además de eso, implementar un algoritmo que sea capaz de rastrear una persona con un teléfono móvil.

Abstract

The goal of this project is based on the characterization of Geofence technology as a location system in Android. In the absence of no real information about the behavior of this system, a first step is taken to test different situations to be able to understand how Geofence acts in practice, and in addition, implement an algorithm that is able to track a person with a mobile phone.

Índice general

Capítulo 1

Introducción, contenido y estructura de la memoria

1.1. Introducción

En los últimos años, el mercado de los teléfonos móviles ha experimentado una gran evolución, destacando el año 2008 con la aparición del primer teléfono móvil con sistema operativo Android, el cual hoy en día está presente en poco más del 80 % de los dispositivos móviles [1].

La abundante cantidad de teléfonos móviles que se encuentran actualmente hacen posible que surja el interés a la captación de datos de carácter personal. Este tipo de datos se definen como cualquier información que haga referencia a una persona física, y son utilizados para la elaboración de diferentes perfiles, con el fin de conocer o predecir los tipos de comportamiento de las personas, sus zonas de interés o incluso por dónde se mueven. Sin embargo, para poder obtener este tipo de datos, previamente es necesario tener el conocimiento de cómo adquirirlos a través de un dispositivo móvil actual de una forma fiable y sin costes.

Este proyecto es presentado como Treball Fi de Grau relacionado con la especialidad de Tecnologies de la Informació de la Facultat d'Informàtica de Barcelona (Universitat Politècnica de Catalunya) en colaboración con el grupo de Investigación SANS[1] del departamento de Arquitectura de Computadores. El propósito del grupo de investigación es la captura y el análisis estadístico de los datos obtenidos por dispositivos de Internet de las Cosas, o en inglés, Internet of Things (IOT).

El objetivo principal de este proyecto es la captura de datos de localización a partir de una implementación software basada en la tecnología Geofence. Gracias a ello, se caracterizará este sistema viendo en detalle cómo se comportan los

[1] Statistical Analysis of Networks and Systems - http://sans.ac.upc.edu/

diferentes componentes de localización y cómo éstos actúan sobre el teléfono móvil.

1.2. Problemas a resolver

Los siguientes puntos son los principales problemas que se quieren resolver durante el desarrollo de este proyecto:

- **Obtención de datos de localización geográfica en segundo plano.** Posibilidad de obtener una localización geográfica a través de un teléfono móvil con conexión a Internet y con disponibilidad del sensor GPS. Como se detalla más adelante, la localización en segundo plano ha sufrido muchas limitaciones en los últimos años a causa de su efecto negativo en la batería de los teléfonos móviles.
- **Escasez de información de la tecnología Geofence.** El software de obtención de los datos de localización debe estar basado en la tecnología Geofence como requerimiento de este proyecto. La problemática que encontramos es que, por lo novedoso que es este sistema, hay poca información pública de cómo es su comportamiento en la práctica. Deberemos caracterizar esta tecnología a partir de diferentes experimentos definidos durante el proyecto.
- **Inexistencia de un algoritmo de rastreo basado en Geofence.** Ser capaz de desarrollar una aplicación que pueda rastrear tus movimientos. Esta implementación software será basada en la tecnología Geofence y deberá ejecutarse en segundo plano.

1.3. Soluciones propuestas e implementadas

A continuación, se presenta un resumen de las soluciones implementadas para resolver cada uno de los problemas anteriores. El detalle de su desarrollo y análisis se especifica en cada uno de los capítulos correspondientes.

- **Obtención de datos de localización geográfica en segundo plano.** La solución a este problema empieza realizando un estudio para caracterizar los diferentes sistemas de localización en Android. Gracias a esta caracterización, se han podido analizar las diferentes dificultades de obtener localización en Android cuando hablamos de una aplicación que se ejecuta en segundo plano.

  Pero finalmente, después de muchas pruebas, hemos visto que la aplicación implementada, solamente funcionaba con la restricción de ubicación en segundo plano deshabilitada. En detalle en los capítulos 5, 6 y 7.

- **Escasez de información de la tecnología Geofence.** Se ha diseñado e implementado un software capaz de monitorizar el comportamiento de Geofence, tanto en ejecuciones en primer plano como en segundo plano, obteniendo datos de localización a partir de transiciones.

  Además, se han definido diferentes escenarios para ver cómo realmente funciona Geofence, y así poder proceder a un análisis de su funcionamiento. Finalmente, se han definido diferentes puntos donde se explican los diferentes comportamientos que tiene Geofence delante de diferentes factores, tanto internos como externos de la aplicación. En detalle en el capítulo 6 y 7.

- **Inexistencia de un algoritmo de rastreo basado en Geofence.** Se ha desarrollado un algoritmo capaz de rastrear un teléfono móvil basado en Geofence. La aplicación que contiene este algoritmo es ejecutada en segundo plano, y para poder probar que realmente funciona, se han definido diferentes modos de ejecución para ver sus limitaciones. En detalle en el capítulo 8.

1.4. Estructura de la memoria

En esta sección se detallará la estructura que sigue la memoria para explicar lo que se ha resuelto durante el desarrollo del proyecto.

Cada uno de los capítulos que se definen a continuación dependen de la ejecución anterior, y es por eso que su explicación se encuentra ordenada cronológicamente, de más antiguo a más reciente.

La primera parte corresponde al contenido de la asignatura Gestió de Projectes, haciendo que las 4 restantes pertenezcan a la parte del desarrollo del proyecto. En primer lugar, se presenta una caracterización de los diferentes sistemas de localización en Android, para que, gracias a esta explicación, ser capaz, en la siguiente sección de implementar un software de monitorización basado en Geofence. Después, se detalla la explicación de cómo se van a capturar datos de localización con el software de monitorización, y entretanto, se puntualiza cada uno de los diferentes experimentos para proceder a su análisis. Por último, se implementa un algoritmo basado en Geofence capaz de rastrear en segundo plano.

Detalladamente, el contenido de cada uno de los capítulos es el siguiente:

1.4.1. Motivación, gestión del proyecto y sostenibilidad

Estos capítulos corresponden a la asignatura Gestió de Projectes. En el capítulo *Motivación y contextualización*, se verá el contexto en el que se desenvuelve el proyecto, y se presenta también, la formulación del problema y el

estado del arte. Seguidamente, en el capítulo de *Alcance, metodología y planificación*, se presenta cuál es el alcance del proyecto, definiendo así la metodología a seguir con un análisis de tiempos y costes económicos del desarrollo del proyecto. Finalmente, en el capítulo *Sostenibilidad*, se hace un análisis del impacto social, ambiental y económico del proyecto.

Todo esto corresponde a los 3 primeros capítulos del trabajo, y se define como el apartado preliminar realizado antes del desarrollo del trabajo.

1.4.2. Caracterización de sistemas de localización en Android

En este capítulo *Caracterización de sistemas de localización en Android*, se definirán cada uno de los proveedores de ubicación en Android y cómo éstos son capaces de obtener una ubicación. Además, se especificará la necesidad de utilizar la tecnología Geofence, y se procederá a elegir la mejor arquitectura software para implementar esta tecnología, y así, cumplir los objetivos de este proyecto.

1.4.3. Software monitorizado basado en Geofence

Después de la elección de la arquitectura que se va a implementar, en este capítulo se detallarán los diferentes pasos que se han hecho para lograr un software capaz de monitorizar la tecnología Geofence. Para ello, primeramente, se ha diseñado el modelo que se quiere implementar, empezando con el desarrollo de dos aplicaciones diferentes: Una en segundo plano y otra del sistema Geofence. Al conseguir esto, finalmente, se procede a la integración de los dos servicios para acabar con una aplicación capaz de recolectar datos de localización de Geofence.

1.4.4. Captura de datos, experimentación y análisis

En el capítulo de *Captura de datos, experimentación y análisis*, se definen los diferentes campos de los datos que se van a obtener y en qué formato van a ser almacenados. Además, se presentarán 10 experimentos a ejecutar, los cuales presentan escenarios completamente distintos, para que finalmente, se analicen y se comprenda el verdadero comportamiento de Geofence.

1.4.5. Desarrollo del algoritmo de rastreo

Para concluir el proyecto, se verá la solución implementada de un algoritmo basado en Geofence, capaz de rastrear un teléfono móvil cuando se ejecuta en segundo plano. Además, se ha implementado el servicio de Push Notification,

para que cuando le llegue a la aplicación un mensaje de notificación, el algoritmo empiece a ejecutarse dentro del teléfono móvil como un servicio en segundo plano y empiece a rastrear.

Capítulo 2

Motivación y contextualización

2.1. Motivación y formulación del problema

El proyecto que se presenta está basado en el término del Big Data, el cual describe la gran cantidad de datos para analizarlos y examinarlos con el fin de llegar a una información útil. Pero no es la cantidad de datos lo más importante en el Big Data, sino que lo que realmente importa son las conclusiones obtenidas a partir de todos los datos que dispones [2].

Principalmente, este proyecto ha sido planteado por la escasa información pública del verdadera comportamiento en la obtención de datos de localización en sistemas basados en Geofence. Hoy en día, hay una tendencia creciente a adquirir estos datos, que sean fiables y que el consumo de batería del dispositivo móvil en su obtención sea mínimo.

Desgraciadamente, la captura de la ubicación en aplicaciones es uno de los principales problemas en la batería de un teléfono móvil [9], y es por eso por lo que mucha gente desactiva la opción GPS de su teléfono, ya que lo asocian a la descarga rápida de batería. Debido a esto, las nuevas versiones Android (a partir de Android Oreo 8.0) han incorporado una restricción en la obtención de este tipo de datos en comparación a las antiguas versiones, limitando la frecuencia en la obtención de datos de localización a solo algunas veces por hora [7].

Por consiguiente, Google te recomienda que para saltarte dicha restricción debes de utilizar una tecnología llamada geofence. Obtener datos de localización a partir de eventos de transición de una área circular previamente definida. Con esto consigues aumentar la frecuencia de las actualizaciones de ubicación, pasando de solo algunas veces por hora a una frecuencia aproximada de dos minutos [7].

Los datos que queremos obtener serán extraídos de una implementación software instalada en un teléfono móvil con sistema operativo Android. Este tipo de datos están recogidos en Ley Orgánica 15/1999, de 13 de diciembre, de

Protección de Datos y en el Reglamento (UE) 2016/679 como datos de carácter personal, algo que explicaremos con detalle más adelante.

Entender cómo funciona esta tecnología es uno de los principales objetivos de este proyecto y gracias a su ejecución ayudará a obtener datos de ubicación más precisos con una mínima descarga de batería en los dispositivos móviles Android actuales.

Por otra parte, el origen real de la propuesta de este proyecto, viene por parte del grupo de investigación donde estoy actualmente trabajando, los cuales están interesados en experimentar el funcionamiento de las Geofences bajo distintos escenarios, y así publicar distintos artículos de investigación para hacer pública dicha información.

2.2. Estado del arte

Android hace posible una comunidad de desarrollo de aplicaciones móvil por el hecho de ser de código libre, es decir, que incluye numerosas API's (métodos que ofrecen diferentes librerías para ser utilizada por otro software como una capa de abstracción) [3], donde todas ellas tienen una licencia y distribución completamente gratuita, haciendo que cualquier dispositivo móvil pueda trabajar con dicho sistema operativo [4].

Gracias a estas API's, se facilita el desarrollo a bajo nivel de Android, haciendo así más fácil la comunicación entre hardware y software del dispositivo. Por ejemplo, nos proporciona el acceso a mensajes entre aplicaciones, procesos (IPC); procesos en segundo plano, uso de mapas, control de multimedia, incluyendo la cámara y el micrófono; y el acceso al hardware de Wifi, Bluetooth, sensores, GPS... Google proporciona una API para obtener la localización, la cual permite a los desarrolladores de aplicaciones solicitar la ubicación prácticamente en cualquier momento [5].

La investigación se centrará en la captación de datos de ubicación en segundo plano, es decir, proceso que se encuentra en ejecución oculto para el usuario. Haciendo esto nos permite la captación de datos mientras el teléfono está trabajando en otras tareas o simplemente esté bloqueado por el usuario, algo que con una aplicación en primer plano, foreground en inglés (visible para el usuario), no se podría conseguir.

2.2.1. Estado actual.

La ubicación en segundo plano o en inglés background es identificada, como se ha comentado anteriormente, el principal problema en la descarga de la batería de los dispositivos móviles. Android, con la aparición de la versión Oreo en 2017, ha introducido límites en la obtención de datos basados en la ubicación en aplicaciones que se ejecutan en segundo plano. La localización en este

caso estará disponible unas pocas veces por cada hora con el fin de disminuir el consumo de batería [7].

Es evidente entonces, que tener un dato de ubicación unas pocas veces por cada hora es muy limitado e insuficiente. Y es por eso que, Google te recomienda la implementación de una tecnología llamada Geofence, a partir de una API que te proporcionan ellos. Gracias a la utilización de esta tecnología, desaparece la restricción de la escasa frecuencia de obtención de localización en aplicaciones en background [7]. Geofence se define como una región circular ubicada en cualquier punto del mapa dada una longitud, latitud y radio; y te notifica cuando el dispositivo móvil entra, sale o permanece dentro de la región del Geofence durante un cierto tiempo.

Resulta oportuno comentar que, existen dos maneras para obtener la ubicación cuando se utilizan aplicaciones Android:

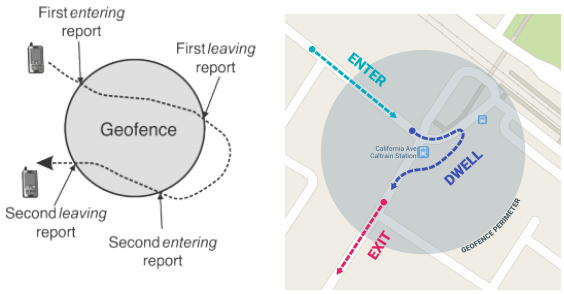
- Framework location, dicho de otra manera Android Location Service. Es la más antigua y se caracteriza por poder obtener la localización proporcionada por el componente de hardware definido previamente. Este tipo de localización no está para nada optimizada en el consumo de batería, y solo podrías obtener una localización siempre y cuando ese componente esté disponible [8].

- Fused location, o también llamado Google Play Location Service. Es una API basada en Google Play Service altamente optimizada en el consumo de batería y muy recomendada por Google. El Fused Location utilizará sensores del dispositivo, tales como acelerómetro, magnetómetro y giroscopio, que juntamente con el GPS o Wifi, obtendrán la localización del teléfono móvil [5].

La tecnología geofence podría ser una herramienta muy importante en cualquier aplicación basada en la geolocalización en un futuro cercano por las numerosas novedades que contiene. De hecho, juntamente con Fused location, nos proporciona un servicio de geolocalización de bajo consumo de energía en aplicaciones ejecutándose en segundo plano.[9].

Este sistema tiene presente la restricción de no poder crear más de 100 Geofences, ya que mantenerlas actualmente es muy caro si hablamos de rendimiento para el dispositivo. Sin la presente restricción, los escaneos constantes de ubicación harían aumentar el consumo de batería en el dispositivo [9]. Esto sería un problema para el principal propósito de la investigación, ya que los datos obtenidos solo se podrían lograr de 100 ubicaciones distintas.

Por lo tanto, uno de los principales desafíos en este tipo de escenario será cambiar las Geofences de posición, con la frecuencia con la que lo hacen los teléfonos móviles. Es decir, ir eliminando esas Geofences que sabes que no volverás entrar y crear otras en que es mucho más probable que salte una notificación. Hablamos del concepto de Geofences dinámicas [11].



(a) (b)

Figura 2.1: Eventos de transición en Geofence. [Figura a] Location events for triggered services. Recuperado de [17]. [Figura b] Create and monitor geofences. Recuperado de [18].

El comportamiento de este sistema depende de la configuración de los parámetros que te proporciona su API. Como por ejemplo fijar el tiempo de notificación que permaneces dentro de un área definida, el tiempo de expiración de una Geofence e incluso el tiempo de aviso cuando salta una notificación. Otro factor fundamental de esta tecnología es como definas los parámetros de configuración del Fused location, el cual te permite definir la frecuencia de tiempo en que solicitas una localización [14].

2.2.2. Marco legal de protección de datos

Este proyecto se basa en el tratamiento de datos de ubicación de los dispositivos móviles, y es por eso que creo que debería haber una sección que tratara sobre el marco legal de la obtención y tratamiento de estos datos.

Antes de empezar, cabe destacar lo que considera el reglamento de la Unión Europea que debe aplicar su marco legal a los datos de carácter personal:

> Los principios de protección de datos no deben aplicarse a la información anónima, es decir información que no guarda relación con una persona física identificada o identificable, ni a los datos convertidos en anónimos de forma que el interesado no sea identificable, o deje de serlo. En consecuencia, el presente Reglamento no afecta al tratamiento de dicha información anónima, inclusive con fines estadísticos o de investigación.
>
> (Consideración 26 del Reglamento (UE) 2016/679)

Todos los datos de ubicación son considerados por la Boletín Oficial del Estado[1] como datos de carácter personal. Este tipo de datos son objeto de

[1] https://www.boe.es/

análisis y evaluaciones para la elaboración de diferentes perfiles. En concreto, los datos de ubicación se podrían analizar para elaborar perfiles de personas, basados en sus movimientos, comportamientos, intereses, salud, entre otros.

Todo esto está muy regulado por la Unión Europea en el Reglamento (UE) 2016/679 y que recoge el Boletín Oficial del Estado en la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales[2]. Su propósito es garantizar la privacidad y la protección de datos. En particular, el reglamento formulado por la Unión Europea no sólo afecta a empresas con domicilio social en Europa, sino que se amplía a empresas que realizan tratamientos de datos de ciudadanos europeos, como por ejemplo Amazon y Google Analitics [12].

Deber de información y consentimiento

El tratamiento de estos datos que se obtienen tanto de aplicaciones o proveedores de servicio deben reunir de una serie de requisitos[12][13]:

- Tener el consentimiento previo del usuario.
- Advertir de los fines de la manipulación de sus datos y la medida en que van a ser tratados.
- Facilidad de omisión del consentimiento del usuario en cualquier momento.
- Poder saber el intervalo de tiempo en que se conservarán los datos de carácter personal y el conocimiento de la identidad del responsable de la gestión.

Seguridad del tratamiento

El Artículo 32 del Reglamento (UE) 2016/679, recoge medidas técnicas y organizativas para garantizar un nivel de seguridad adecuado de los datos personales:

- Cifrado de los datos y aplicación de la seudonimización. Es decir, dificultar la acción de vincular una persona física con su datos. Para ello, se utilizas técnicas de cifrado, por ejemplo cifrado con clave secreta, función de hash, descomposición en tokens, entre otros métodos.
- En caso de accidente físico o técnico debe haber una capacidad de restaurar la disponibilidad al acceso de los datos de forma rápida.
- Establecer métodos de copias de seguridad.
- Un proceso de verificación para garantizar la seguridad del tratamiento.

[2]https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673

- Ser capaz de garantizar la confidencialidad y disponibilidad de los sistemas y servicios de tratamiento.
- En caso de violación de seguridad, el responsable debe notificarlo tanto a la autoridad de control(autoridades públicas) como a los interesados como máximo 72 horas de tener constancia de la intrusión. Se deberá describir la naturaleza del ataque a la autoridad y comunicar las posibles consecuencias a ambos.

Ejemplos de artículos interesantes

A continuación se pondrán resumidos algunos de los artículos que he encontrado más interesantes sobre la manipulación de los datos de carácter personal.

> Todo afectado por la obtención de datos de carácter personal se le proporcionará si desea una información básica que deberá contener, al menos la identidad del responsable del tratamiento y la finalidad del tratamiento. Cuando los datos personales no han sido obtenidos del afectado se deberá incluir: Las categorías de datos objeto de tratamiento y las fuentes de las que procedieran los datos.
>
> (Artículo 13 del Reglamento (UE) 2016/679)

> El tratamiento de los datos no deben provocar daños y prejuicios físicos, materiales o inmateriales. Más concreto que su tratamiento puedan dar lugar a problemas de discriminación, usurpación de la identidad o fraude, daño para la reputación.
>
> (Articulo 9 de la Ley Orgánica 3/2018)

Se podría decir que los datos son considerados como el oro del siglo XXI para las empresas y es por eso que su adquisición es algo en que se está apostando hoy en día. Gracias a su análisis, estas empresas, podrían tener una información que podría llegar a tener un carácter privado para las personas. Por ejemplo si a ti te asignan en un perfil apropiado, podrían predecir con poco margen de error cuáles son tus movimientos o cuáles son tus comportamientos. Es por eso que creo que es importante poner unos limites en su manipulación y que las sanciones sean duras para quién no cumpla con las leyes establecidas.

2.2.3. Estudios y aplicaciones existentes.

- **Geofencing for Fleet & Freight Management** [15]**.**

Es un artículo científico que trata sobre la utilización de la tecnología Geofence en el sector de transporte y logística. Te da una explicación sobre esta tecnología y cómo poder utilizarla. Por ejemplo presenta técnicas de rastreo en aplicaciones basadas en Geofence. Como la mayoría artículos científicos de Geofence, no detalla en sus explicaciones, pero es interesante ver las diferentes funciones que puede tener esta tecnología tan novedosa.

- **Aplicaciones basadas en POIS.**

  Muchas son las aplicaciones dónde implementan la tecnología Geofence con POIS, puntos de ubicación de interés. Es decir, aplicaciones que definen Geofences estáticas en puntos definidos por el usuario. Por ejemplo, si la Sagrada Familia fuera un punto de interés para un usuario, la aplicación construiría una Geofence a su alrededor y notificaría al usuario en caso de transición.

- **Aplicaciones control parental: Spyzie** [16]**.**

  Existen aplicaciones basadas en Geofence instaladas en los dispositivos móviles de los niños. Se fundamenta en la definición de diferentes áreas de control, para que cuando un niño entra o salga de esa área se notifique a los padres. Parecido a nuestro proyecto, ya que mantienes el control de la localización de un teléfono móvil. A partir de las diferentes Geofences definidas como áreas de control, se podría llegar a un estudio para rastrear la ruta del niño.

2.2.4. Actores Implicados.

Actualmente el análisis y la captura de datos son importantes en muchos sectores, y si hablamos de datos de localización de personas todavía más. Los actores implicados en este proyecto deberían estar interesados en cómo obtener datos fiables de ubicación de teléfonos móviles Android, ya que tenerlos te da una posición privilegiada en muchos ámbitos, tanto de publicidad, investigación o ámbito social. A continuación, se detallará una serie de actores que podrían estar interesados en el desarrollo de este proyecto.

- **Empresas y/o Organizaciones.**

  Cada vez son más las empresas que se interesan en el análisis de datos de ubicación de personas, con el fin de llevar a cabo diversos estudios de sus movimientos y así obtener beneficios. Por ejemplo, en el mundo de la publicidad, es muy importante saber la densidad de personas en un punto determinado, ya que el incremento de campañas publicitarias en esas zonas, podría hacer aumentar las ventas del producto o servicio que se da a conocer. Cuanta más gente lo vea más impacto tendrá en las ventas. Es por eso por lo que muchas empresas podrían estar interesadas en este proyecto, necesitan capturar datos para poder sacar conclusiones y darles una utilidad.

Entender cómo funcionan o actúan los diferentes componentes de localización en un teléfono móvil, podría dar a las empresas u organizaciones otras visiones para el desarrollo de aplicaciones basadas en la obtención de datos. Escoger una buena estrategia en la captura de estos datos, puede darte muchas ventajas en comparación a otras empresas del mismo sector.

- **Grupo de investigación SANS.**

  Grupo de investigación donde actualmente estoy trabajando, interesado en cómo obtener datos de localización fiables y tener el conocimiento del comportamiento de los elementos del teléfono móvil que te proporcionan una ubicación. Lógicamente, todos los miembros formantes del grupo están interesados en la evolución de este proyecto, debido a que la información pública sobre este tema escasea actualmente, sobretodo cuando hablamos de la tecnología Geofence. Los principales interesados en este proyecto son los coordinadores, Jorge García Vidal y José Ma. Barceló Ordinas.

- **Personas implicadas.**

  Lógicamente, no hay datos si no hay gente con una aplicación de geolocalización instalada en un teléfono móvil. Por este motivo, toda esa persona que tenga aplicación de recogida de ubicación debería ser considerada como un actor implicado más. Los datos obtenidos de estas personas serán objeto de abundantes estudios y de análisis exhaustivos.

Capítulo 3

Alcance, metodología y planificación

3.1. Alcance del proyecto.

Para encontrar la mejor solución posible, debe haber un periodo de aprendizaje del funcionamiento de Android en base a la localización. Saber cómo es capaz el teléfono de obtener su ubicación y entender cuál es su comportamiento en segundo plano es fundamental para empezar a plantearte diferentes arquitecturas en el software de la aplicación.

La existencia de una buena implementación en segundo plano, sabiendo las restricciones que conlleva en los nuevos teléfonos móviles, permitirá obtener datos fiables con el fin de analizarlos. Además, para poder implementar esta característica, es necesario tener claro todos los componentes de aplicación de Android, tales como Activity, View, Service, Intent, etc.

Una vez que se construya la aplicación, pasaremos a la siguiente fase de obtención de datos y de la búsqueda de posibles errores en el código. Para almacenar los datos, se configurará una base de datos en Amazon Web Servicie, llamada DynamoDB. Si fuera el caso de encontrarnos la situación de pérdida de datos, se decidirá guardarlos en un fichero del teléfono para que sea más fiable y mejor accesible.

Al ser un proyecto de experimentación, los diferentes datos de localización obtenidos, serán analizados con el fin de encontrar la mejor arquitectura software. Este procedimiento llevará un tiempo, ya que dependes de muchos factores. Primero necesitaríamos tener el conocimiento del comportamiento de la tecnología Geofence, ya que actualmente hay muy pocas publicaciones sobre cómo se comporta ejecutándose en segundo plano. En segundo lugar, para hacer posible la generación de datos, necesitarías que alguien saliera a dar una vuelta por la calle y recorriera una ruta especificada. Para validar los datos obtenidos, se

comparará la ruta definida con los datos obtenidos de la aplicación.

A continuación, se implementará diferentes algoritmos de rastreo basados en tecnología Geofence, con el fin de saber los movimientos de una persona sin el conocimiento de su ruta. Además, debemos corregir todo aquello que no ha salido como esperábamos o mejorarlo.

Una vez encontrada la implementación correcta, llega otra fase de análisis de datos, validando que la ruta que se ha hecho es correcta con los datos obtenidos. Llegar a conclusiones de como el dispositivo utiliza la tecnología Geofence y como es capaz de utilizar geolocalización con una descarga baja de batería. Cuantificar los datos y ver en qué punto es coherente con la prueba de diferentes variables que te proporciona el teléfono, como el nivel de rendimiento del teléfono, el porcentaje de batería, el estado del GPS o el proveedor de localización.

En resumen, a partir de una correcta implementación software de una aplicación Geofence, se deberán encontrar diferentes algoritmos de rastreo y a la vez entender cómo el dispositivo móvil utiliza la tecnología Geofence para tener un mejor rendimiento.

3.1.1. Objetivos y requerimientos.

Como se ha comentado anteriormente, este proyecto tiene como objetivo caracterizar el sistema Geofence a partir de datos obtenidos de una implementación software capaz de monitorizar dicha tecnología. Para lograr el objetivo principal se tiene que seguir los siguientes objetivos y a la vez requerimientos:

- Caracterizar los diferentes sistemas de localización de android.
  - Tener el conocimiento cómo es capaz un teléfono móvil de obtener una ubicación con tan poco margen de error y cómo puede afectar al dispositivo en su obtención.
- Desarrollo de una implementación software basada en Android capaz de monitorizar la tecnología Geofences.
  - A partir de la monitorización, se requiere obtener datos de localización fiables.
  - Se deberán conocer con detalle los diferentes componentes software de las aplicaciones para llevar a cabo un servicio en segundo plano.
  - Conocer las diferentes APIs para la obtención de ubicación y cómo funcionan.
- Hallar la mejor configuración tanto del proveedor de ubicación como de la tecnología Geofence para su correcto funcionamiento ejecutándose en segundo plano.

  - Experimentar diferentes maneras de desarrollo software a partir de los datos obtenidos de localización. Se deberá ver cuáles son las distintas configuraciones en los parámetros del proveedor de ubicación y de la tecnología Geofence. Además se deberá escoger una buena arquitectura para construir un servicio en segundo plano que soporte las actualizaciones de localización y el sistema Geofence.
  - Caracterizar los diferentes factores que pueden intervenir en la obtención de la localización. Ver cómo diferentes variables pueden afectar a la obtención de datos, como por ejemplo la batería que se disponga, estado GPS o el tiempo de planificación del servicio en segundo plano.
  - Cuantificar la información proporcionada por la tecnología Geofence. Analizar la lista de avisos y ver en qué punto puede ser coherente por cada experimento.

- Tener el conocimiento de la ubicación de un teléfono móvil durante un trayecto no definido anteriormente.

  - Desarrollo de algoritmos de rastreo para seguir los diferentes movimientos de una persona, a partir de una aplicación instalada en su teléfono móvil.

3.1.2. Riesgos y posibles obstáculos.

Nacimiento de una nueva versión Android no compatible. Se podría dar el caso del surgimiento de una versión Android sin la compatibilidad con la tecnología Geofence. Es algo que actualmente es poco probable, ya que Google ha apostado muy fuerte por esta tecnología como una de las soluciones de geolocalización en segundo plano. Debemos tener en cuenta este riesgo, debido a que en un futuro podría surgir otra tecnología que haga que Geofence desaparezca.

Interferencias con otras aplicaciones. El teléfono móvil te permite tener varias servicio ejecutándose en segundo plano. Esto nos lleva a la conclusión de que el teléfono contiene implementada una cola de prioridades dónde están todos los procesos presentes, esperando a ser ejecutados por la CPU. Esto es un entorno que hoy en día no podemos controlar. El caso es que, podemos tener la situación de que nuestra tarea en segundo plano puede ser olvidada por la CPU por las numerosas tareas con mayor prioridad.

Una solución para este problema sería tener un teléfono móvil con las aplicaciones básicas para tener un entorno controlado. Esta solución no sería válida en la práctica, ya que cualquier teléfono de hoy en día contiene numerosas aplicaciones en segundo plano. Buscar una solución para este posible problema será un desafío para el desarrollo del proyecto.

Errores de código. Este contratiempo es bastante probable, y encontrarlo llevará un tiempo, ya que para examinarlo necesitamos salir a la calle para ver si

la solución solventa el error. Esto es algo que conlleva bastante tiempo. Además, cabe decir que, las soluciones a los diversos errores o problemas se debatirán en las reuniones semanales, buscando así su mejor resolución.

Problemas de conectividad. Para tener datos de localización en Android necesitas un proveedor, en este caso como se ha comentado anteriormente, el Fused Location. Este proveedor se caracteriza por obtener la localización a partir de la combinación de múltiples sensores con la tecnología de localización (GPS, Wifi o células telefónicas).

En la práctica, hay momentos en que la conectividad podría ser bastante mala o nula, y es por eso por lo que se pueden obtener datos no coherentes o inexistentes. Esto será un reto en el apartado de la creación de algoritmos de rastreo, ya que la presencia de una zona sin conectividad, podría hacer perder el seguimiento de un teléfono móvil. Veremos si este obstáculo estará presente y se verá cuál podría ser su posible solución.

Aparte de esto, también se puede dar el caso de problemas de conectividad con nuestra base datos. Amazon Web Services contiene diferentes herramientas para visualizar registros y encontrar el problema debería ser bastante rápido, para así dar con una solución.

3.2. Metodología y rigor.

3.2.1. Métodos de trabajo.

Tener una metodología de trabajo claramente definida debe ayudar a tener el control del proyecto y reducir el número de obstáculos que puedas encontrarte durante su desarrollo. Además, ayudará a ahorrar tiempo y tener una capacidad de respuesta a los posibles cambios.

Actualmente en el desarrollo de software se implementa una metodología ágil, la cual tiene un enfoque en proyectos de software basados en el desarrollo iterativo e incremental. Este tipo de metodología tiene el fin de ser muy ágil y flexible en el desarrollo de productos con una buena calidad en el resultado final.

Para el desarrollo de este proyecto se utilizará el módulo SCRUM basado en una metodología ágil. Es la técnica más empleada actualmente en el desarrollo software y es la que mejor que se adapta en este tipo de proyecto. Aportará mayor control en el desarrollo del proyecto y en temas de análisis de datos. También ayudará a organizar y definir las diferentes tareas a realizar a partir de los objetivos establecidos.

En primer lugar, diferenciamos dos elementos, los actores y las acciones. Lógicamente, los actores serán quienes ejecuten las acciones. Tenemos como actores:

- Usuarios: los beneficiarios del proyecto.
- Equipo SCRUM: desarrolladores del proyecto. En este caso el autor del trabajo.
- ScrumMaster y dueño del producto: en este caso serán el director y co-director de este proyecto. Con el objetivo de velar el cumplimiento de la metodología y guiar al desarrollador a cumplir su meta. A parte de esto, previamente al comienzo el proyecto, han sido ellos quienes han marcado los requerimientos.

Las acciones se dividen en diferentes categorías, según el estado de desarrollo que tenga. Una acción que aún no se ha empezado a realizar será puesta en la categoría de Backlog. En esta categoría tendremos esas acciones a realizar y los objetivos que se pretenden conseguir, marcado todo por el dueño del producto y ScrumMaster. Estas tareas se irán moviendo según su estado, "en proceso" o "finalizado".

Una vez a la semana se llevará a cabo un Sprint Planning Meering. Reunión que sirve para decidir y planificar qué tareas se van a hacer durante la semana. Antes de hacer esto, se comentarán los diferentes resultados de las tareas finalizadas de la semana anterior a la realización del Sprint Planning Meering. Si una acción no es acabada se buscaría la razón por la cual no ha sido finalizada; se hablaría de los obstáculos encontrados y sus posibles soluciones.

3.2.2. Herramientas de seguimiento.

Las principales herramientas que se utilizar para garantizar el resultado del proyecto será Trello[1] y Git[2].

Trello es un software de administrador de proyectos que te permite la creación de tickets donde se definen las diferentes acciones a realizar. Gracias a ella se llevará el control de lo que tienes hecho y de lo que te queda por hacer. El seguimiento del proyecto será posible haciendo que los diferentes tickets definidos vayan pasando de una categoría a otra: backlog, en proceso y finalizado. Cada ticket contiene un sistema de comentarios donde se pondrán cada uno de los detalles de la tarea asignada.

A parte de Trello, para poder llevar a cabo el desarrollo software de la aplicación se utilizará Git, un controlador de versiones software que facilitará tener el control de los diferentes cambios software en la aplicación. Se utilizará GitLab para la administración del repositorio del proyecto y de sus diferentes ramas, manteniendo así el desarrollo software seguro.

Finalmente se utilizará Microsoft OneNote[3] como herramienta supletoria y

[1] https://trello.com
[2] https://git-scm.com/
[3] https://products.office.com/es-es/onenote/

de soporte. Servirá para apuntar las diferentes ideas que vayan surgiendo durante el desarrollo del proyecto con el fin de etiquetarlas y compartirlas con el director.

3.2.3. Métodos de validación.

Los datos obtenidos en el desarrollo de este proyecto no podrán ser objeto de la comparación con otros valores conocidos, ya que la tecnología Geofence es un sistema tan novedoso que toda persona que ha trabajado con ello, no ha detallado la fiabilidad de los datos que esta tecnología te proporciona. Es por eso que se ha decidido implementar un programa en Pyhton para validar los datos de localización. Este programa genera un html donde mostrará todos los datos de ubicación obtenidos en un mapa con todas las Geofences dibujadas y con ellos se decidirá si los datos tienen sentido o no. Adicionalmente, con este método de validación veremos si hay errores de código, por las incoherencias de los datos mostrados. El mapa resultante de cada obtención de datos será revelado y discutido durante las reuniones semanales.

A parte de esto, a lo largo del proyecto se garantizará una comunicación con el director y codirector, con el fin de facilitar la resolución rápida de las dudas y problemas que vayan surgiendo.

3.3. Planificación y descripción de las tareas

En este apartado se detallará la planificación que se va a llevar a cabo para la correcta realización de este proyecto. Se describirán las diferentes tareas a ejecutar con sus dependencias y se especificará un plan de acción para corregir las posibles desviaciones que se puedan producir. Cabe destacar que la planificación está sujeta a posibles alteraciones según se desarrolle el proyecto, afectando así al tiempo previsto de su realización.

3.3.1. Descripción de las tareas

A continuación, se describen detalladamente todas las tareas que forman parte del desarrollo del proyecto, ordenadas cronológicamente.

Búsqueda y estudio del modelo

Es la primera fase del proyecto justo después de su aprobación. El objetivo de esta etapa es recolectar información sobre el dominio del proyecto y así tener una visión del camino a seguir para realizarlo correctamente. En esta sección será clave conocer exactamente qué es Geofence, cómo funciona y qué ventajas tiene hacia otras tecnologías. Justo después, se estudiará la viabilidad del proyecto a partir de la información encontrada.

Gestión del proyecto

En esta fase se incluye la elaboración de la asignatura de GEP (Gestió de Projectes). El objetivo de esta etapa es que el proyecto tenga claramente definido su contexto, sus objetivos a cumplir y que tenga una buena planificación para su correcto desarrollo.

Para lograrlo la asignatura consta de 4 secciones: alcance del proyecto y contextualización; planificación temporal, gestión económica y sostenibilidad; y presentación oral y documento final. Cada sección tiene un entregable y el documento final integra las tres primeras secciones con la corrección de la retroacción del tutor de la asignatura Esta tarea no se podrá realizar hasta que no se recopile toda la información de la tarea anterior.

Familiarización, análisis general y creación del entorno

El propósito de esta sección es familiarizarse con la tecnología Geofence y el entorno de desarrollo Android; y hacer un análisis general para tener los conocimientos suficientes para empezar el proyecto. Se realizará un estudio del modelo a partir de la información recogida en la primera tarea y de informes encontrados posteriormente. Por último, se procederá a la creación del entorno de trabajo con la instalación de diversos programas.

Diseño e implementación

Esta tarea tiene el objetivo de la creación de una aplicación Android que proporcione datos de ubicación ejecutándose en segundo plano a partir de una implementación basada en la tecnología Geofence. La realización correcta de la tarea anterior será clave para la ejecución de esta, ya que un buen conocimiento del entorno puede ayudar a un correcto diseño e implementación de la aplicación.

Experimentación y corrección de errores

La fase de experimentación se llevará a cabo una vez que la aplicación esté implementada y se pueda instalar en el teléfono móvil. El objetivo será la obtención de datos de ubicación para poder analizarlos y así contabilizar posibles errores para proceder a una mejora de la implementación software. Es la etapa que más horas se dedicarán, ya que para validar el correcto funcionamiento de la aplicación tienes que salir a la calle y darte un buen paseo sin saber que la aplicación funciona realmente. Además, después de cada paseo tienes que averiguar que es lo que funciona realmente y lo que no en la aplicación, y que factores nos están afectando en la obtención de datos de ubicación. Esto será gracias al análisis que haremos en la siguiente sección.

Análisis de datos

El objetivo de esta sección es encontrar diferentes conclusiones finales a partir del análisis de datos y así poder cumplir con los objetivos establecidos en el comienzo del proyecto.

El análisis de datos contiene dos etapas con finalidades distintas. Primeramente, se llevará a cabo un análisis con el fin de mejorar la aplicación con un estudio de los datos obtenidos en la etapa de experimentación. Esta etapa se llevará a cabo a partir de diferentes discusiones en la reunión semanal con el director y codirector del proyecto. Posteriormente, después de la correcta configuración de la aplicación, se hará un análisis con el fin de cuantificar los datos de ubicación obtenidos y así poder ser capaz de desarrollar algoritmos de rastreo basados en la tecnología Geofence.

Algoritmo de rastreo basado en Geofence

Esta sección será la última en el desarrollo del proyecto y no se podrá empezar hasta tener realmente una aplicación que funcione perfectamente ejecutándose en segundo plano. Su propósito es tener el conocimiento de los movimientos del teléfono móvil en tiempo real a partir de los diferentes datos obtenidos.

Para satisfacer el objetivo de esta etapa se precisará de un algoritmo de rastreo capaz de saber la ubicación del teléfono móvil en un momento determinado. La culminación de esta sección será dada por una correcta implementación de una aplicación ejecutándose en estado background, capaz de seguir los movimientos de una persona.

Evaluación de rendimiento

Una vez finalizadas todas las etapas explicadas anteriormente, se hará un análisis y validación de todos los resultados obtenidos durante la realización del proyecto. Por otra parte, tendremos que tener una idea clara y ordenada de todo lo que se ha hecho durante el proyecto para encarar la memoria final.

Memoria final y defensa

Es la última tarea a realizar en el proyecto. Se elaborará una memoria final con el fin de explicar todo lo que se ha hecho durante la realización del proyecto de manera clara y ordenada. Como plan de actuación, se ha decidido que se irá escribiendo en la misma medida que el proyecto vaya avanzando en el tiempo. Con la finalización de la memoria final, se empezará a preparar la presentación oral para la defensa delante de un tribunal.

3.3.2. Previsión temporal

Dedicación

Tarea	**Dedicación (Horas)**
Búsqueda y estudio del modelo	20
Gestión del proyecto	75
Familiarización, análisis general y creación del entorno	35
Diseño e implementación	55
Experimentación y corrección de errores	110
Análisis de datos	100
Algoritmo de rastreo basado en Geofence	50
Evaluación de rendimiento	20
Memoria final y defensa	60
Total	**525**

Cuadro 3.1: Dedicación en horas por cada tarea.

Diagrama de GANTT y dependencias entre tareas

En este apartado se muestra el diagrama de GANTT con la fecha de inicio y final de cada acción. Además, se muestran todas las dependencias que se encuentran entre tareas.

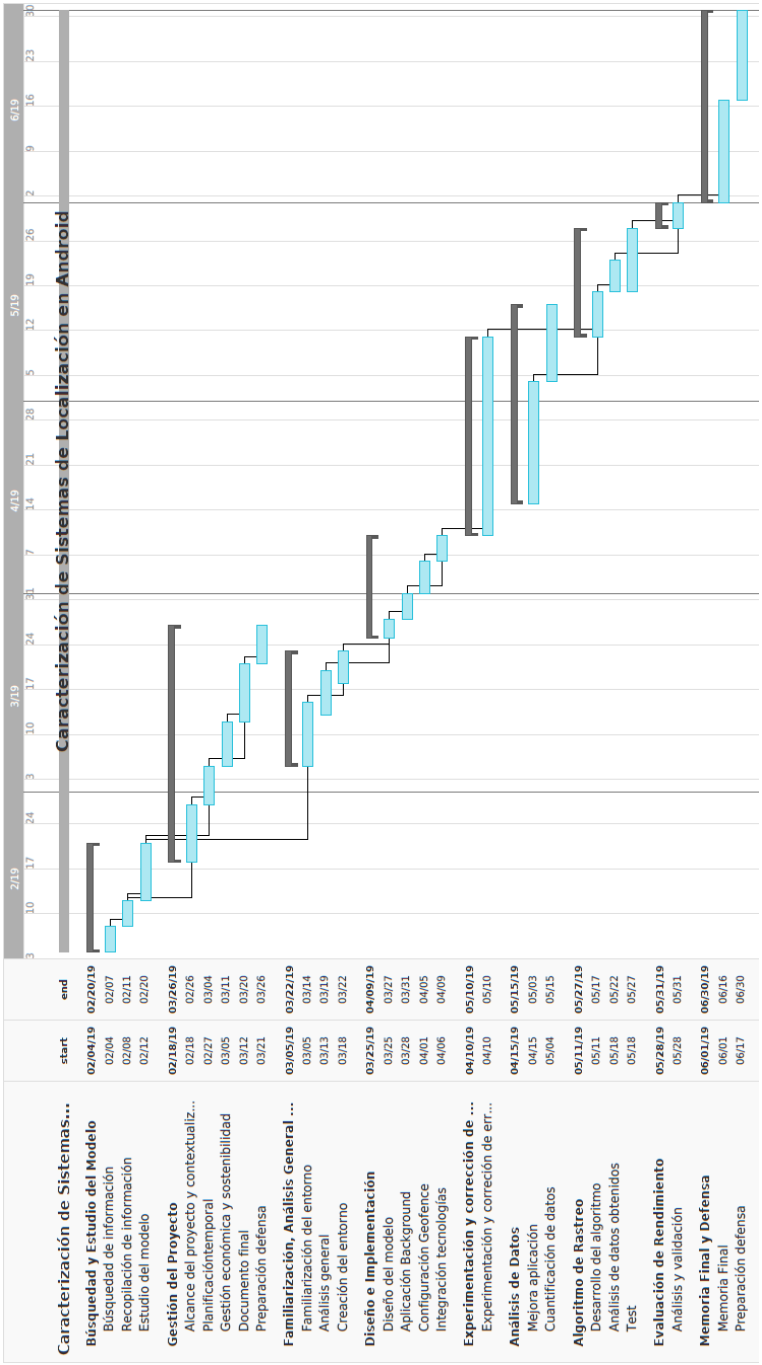


Figura 3.1: Diagrama de GANTT

3.3.3. Recursos

A lo largo del proyecto se utilizarán diferentes tipos de recursos con el fin de un correcto desarrollo. A continuación, se describen los recursos hardware, software y humanos.

Recursos Hardware

- **Ordenador portátil personal HP.** Procesador Intel® Core™ i5-7200U, 8GB de RAM, 128GB de SSD y tarjeta gáfica Intel® HD Graphics 620. Usado para el desarrollo software y la escritura de las memorias del proyecto.
- **Ordenador sobremesa del laboratorio DELL.** Procesador Intel® Core™ i5-6500, 8 GB de RAM, 1 TB HDD y tarjeta gáfica Intel® HD Graphics 530. Su uso es el mismo que el anterior.
- **Móvil Xiaomi Redmi Note 5 personal.** Porcesador Snapdragon 636, 3 GB de RAM y Batería de 4000mAh. Se utilizará para pruebas de la aplicación y la obtención de datos.
- **Huawei P20 lite del grupo de investigación.** Procesador Kirin 659 a 2,36GHz, 4 GB de RAM y batería de 3000mAh. Su uso es el mismo que el anterior.

Recursos Software

- **Sistemas Operativos.** Windows 10 instalado en los dos ordenadores descritos anteriormente, Android Oreo bajo MIUI 9.5 en el móvil Xiaomi personal y Android 8.0 Oreo bajo EMUI 8.0 en el teléfono del grupo de investigación.
- **Control de versiones.** Se utilizará Git y Bitbucket.
- **Editores de texto.** SublimeText para el desarrollo de scripts y del mapa de la validación de los datos de ubicación. Se utilizará Python y JavaScript para su desarrollo.
- **Android Studio.** Entorno de desarrollo de Android basado en el lenguaje de programación Java.
- **Otros.** LaTex para la documentación del proyecto. Excel (CSV) utilizado para dar formato a los datos obtenidos. OneNote para apuntar la información obtenida y TeamGantt para la generación del diagrama de GANTT del apartado anterior.

Recursos Humanos

- **Director y codirector.** Encargados del seguimiento y supervisión para cumplir todos los objetivos establecidos. Ayudaran en la toma de decisiones y en problemas que vayan surgiendo durante el desarrollo del proyecto.
- **Desarrollador del proyecto.** Encargado del desarrollo del proyecto y que se lleve a cabo en las fechas establecidas.

3.3.4. Desviaciones y plan de acción

El desarrollo del proyecto debería acabar a finales del mes de mayo para tener 3 semanas de escritura de la memoria final. Estas 3 semanas podrían verse afectadas por alguna desviación del proyecto y es por eso que se ha decidido escribir la memoria final conjuntamente con el desarrollo del proyecto. De esta forma, se crea un margen de tiempo de maniobra por si surge algún problema.

La desviación más probable, y la única que podemos conocer actualmente, será la puesta en marcha de la aplicación en el teléfono móvil, que corresponde a la etapa de experimentación. Lo que buscamos es que la aplicación a implementar funcione para todos los tipos de móviles basados en Android. Pero surge el problema que muchos teléfonos móviles contienen lo llamado firmware personalizado, que es el caso del teléfono Xiaomi. Este firmware personalizado se encuentra implementado junto el sistema operativo Android y hace que el comportamiento no sea el mismo cuando comparamos diferentes teléfonos móviles. Esto podría afectar en la obtención de datos de ubicación, ya sea por el tiempo de ejecución de la aplicación en background o el tiempo de notificación. Una de las soluciones ya planteadas es tener un teléfono móvil solo con Android instalado, proporcionándonos así un mayor control del entorno dentro del teléfono móvil. Además de esto, se instalará diversos programas de monitorización de procesos en ejecución para controlar cuando se ejecuta nuestra aplicación.

Esta desviación provocará que pasemos la mayor parte del tiempo del desarrollo del proyecto en la etapa de experimentación. Pero gracias a utilizar la metodología Scrum dentro del grupo de investigación nos ayudará a tomar la mejor decisión para este problema y muchos más que podamos encontrar.

3.4. Gestión económica.

Para poder llevar a cabo este proyecto, tenemos que hacer un estudio para determinar si su realización es económicamente viable. Para poder hacer dicho estudio se precisa de un presupuesto total del proyecto y de la descripción de control de gestión delante de posibles desviaciones.

3.4.1. Presupuesto.

El cálculo del presupuesto del proyecto viene dado de diferentes costes: costes directos, costes indirectos, contingencias e imprevistos.

Costes Directos.

Para calcular los costes directos dividiremos los diferentes recursos que se necesitan para la realización del proyecto.

Recursos Humanos

Para ponerse en un contexto real, se muestra a continuación el coste total de los recursos humanos con los sueldos[4] correspondientes a cada rol que se tendrá en el proyecto.

Rol	**Horas Estimadas**	**Salario (€/h)**	**Coste (€)**
Jefe de Proyecto	155	22	3410
Analista	185	12	1860
Diseñador	10	15	150
Desarrollador	155	12	1860
Ingeniero QA	20	18	360
Total	**525**	-	**760**

Cuadro 3.2: Costes Recursos Humanos.

Sin embargo, al realizar yo este proyecto trabajando como jefe de proyecto, analista, diseñador, desarrollador e ingeniero QA[5]; se decide empezar esta sección en referencia al convenio de la Facultat d'Informàtica de Barcelona[20], donde define el salario de un becario como 8€/h.

[4] https://www.indeed.es/salaries
[5] Garantía de Calidad

Rol	**Horas Estimadas**	**Salario (€/h)**	**Coste (€)**
Jefe de Proyecto	155	8	1240
Analista	185	8	1480
Diseñador	10	8	80
Desarrollador	155	8	1240
Ingeniero QA	20	8	160
Total	**525**	-	**4200**

Cuadro 3.3: Costes Recursos Humanos.

Para poder explicar mejor esta repartición de horas, se ha llevado a cabo un estudio de coste por actividad del diagrama de GANTT del apartado *planificación temporal*.

Actividad	**Horas**	**Recursos**	**Coste (€)**
Búsqueda y Estudio del Modelo	**20**		**160**
Búsqueda de información	10	Jefe de Proyecto	80
Recopilación de información	3	Jefe de Proyecto	24
Estudio del modelo	7	Jefe de Proyecto	56
Gestión del Proyecto	**75**		**600**
Alcance del proyecto y contextualización	20	Jefe de Proyecto	160
Planificación temporal	15	Jefe de Proyecto	120
Gestión económica y sostenibilidad	15	Jefe de Proyecto	120
Documento final y preparación defensa	25	Jefe de Proyecto	200
Familiarización, Análisis General	**35**	-	**280**
Familiarización del entorno	20	Desarrollador	160
Análisis general	10	Desarrollador	80
Creación del entorno	5	Desarrollador	40
Diseño e Implementación	**55**	-	**440**
Diseño del modelo	10	Diseñador	80
Aplicación background	20	Desarrollador	160
Configuración Geofence	15	Desarrollador	120
Integración tecnologías	10	Desarrollador	80
Experimentación	**110**	Analista	**880**
Análisis de Datos	**100**	-	**800**
Mejora aplicación	60	Desarrollador	480
Cuantificación de datos	40	Analista	320
Algoritmo de Rastreo	**50**	-	**400**
Desarrollo del algoritmo	15	Desarrollador	120
Análisis de datos obtenidos	15	Analista	120
Test	20	Analista	160
Evaluación de Rendimiento	**20**	Ingeniero QA	**160**
Memoria Final y Defensa	**60**	Jefe de Proyecto	**480**
Total	**525**	-	**4200**

Actividad	Horas	Recursos	Coste (€)

Cuadro 3.4: Coste por actividad.

En esta tabla se puede apreciar la repartición de tareas entre los diferentes roles. Cabe destacar los papeles que se van a hacer de jefe de proyecto, analista y desarrollador. Primeramente, tenemos el jefe del proyecto, el cual se encargaría de toda la planificación y de la redacción de toda la documentación demandada. Después tenemos el desarrollador del proyecto, el cual deberá estudiar todo el modelo software del proyecto y proceder a su implementación. También comentar que como se dijo en la *planificación temporal*, la etapa de experimentación es la que llevará más carga de trabajo. El papel en dicha tarea sería el del analista.

Sumando todas las actividades nos daría un total en costes de recursos humanos de 4200€.

Recursos Hardware

Las horas estipuladas en la siguiente tabla son las horas estimadas de uso de cada dispositivo hardware que se utilizará durante la realización del proyecto. Tenemos en cuenta su vida útil y su amortización.

Producto	Coste	Horas(h)	Vida útil(años)	Amortización (€)
Ordenador personal	699	350	5	5.58
Ordenador laboratorio	599	200	5	2.76
Móvil personal	173	210	2	2.07
Móvil laboratorio	205	180	2	2.11
Total	**1676**	-	-	**12.52**

Cuadro 3.5: Costes Recursos Hardware.

Recursos Software

Cómo se puede ver en la siguiente tabla la mayoría de recursos software son de código abierto, es decir gratuitos. No es el caso del software proporcionado por Microsoft: Microsoft Office y Windows 10 Home. En este caso, también tenemos en cuenta la amortización de cada recurso software que se utilizará.

Costes Indirectos.

Para conocer los costes indirectos del proyecto, hemos cogido de referencia la explicación que hace en el Horizon 2020 para calcularlos [21]. Para poner

Recurso	**Coste**	**Horas(h)**	**Vida útil(años)**	**Amortización (€)**
Sistemas Operativos	**145**	**940**	**-**	**5.58**
Windows 10 Home	145	550	5	1.938
Android	0	390	5	-
Control de versiones	**0**	**525**	**-**	**0**
Bitbucket	0	525	5	0
Editores de Texto	**0**	**65**	**-**	**0**
Sublime Text	0	65	-	0
Android Estudio	**0**	**190**	**-**	**0**
Otros	**69**	**170**	**-**	**0.40**
LaTex	0	120	-	0
Microsoft Office	69	50	1	0.40
Total	**214**	**-**	**-**	**5.98**

Cuadro 3.6: Costes Recursos Software.

en contexto, Horizon es el mayor programa de investigación e innovación en la Unión Europea[22].

Para su cálculo cargan una tarifa única del 25 % de los costes directos. Se aplica una tarifa única por el simple echo de que los costes indirectos no se pueden establecer con exactitud, ya que dependen de muchos factores. Los típicos ejemplos de estos costes serían el consumo energético, el coste del equipo de la oficina, la tarifa de Internet y móvil; y los costes de los terrenos del lugar donde se desarrolla el proyecto.

Entonces, a partir del porcentaje único del 25 %, podemos calcular los costes indirectos de este proyecto como: 4218.5 x 0.25 = 1054.625€

Contingencias.

El nivel de contingencia que se reservará para este proyecto será del 15 % de los costes directos e indirectos. Este porcentaje sale de la incertidumbre, pero cabe decir que no se prevén demasiados problemas durante la realización del proyecto, pero al no tener la certeza de que pasará en un futuro, hay que asegurarse.

Imprevistos.

Puede darse el caso que durante el desarrollo del proyecto surgieran imprevistos o contratiempos. Y es por eso que, las estimaciones anteriores deben estar sujetas a la medición de la incertidumbre como costes en imprevistos, reflejados en el presupuesto del proyecto. Los principales imprevistos serian:

Tipo	**Porcentaje (%)**	**Precio (€)**	**Coste (€)**
Costes Directos	-	4218.5	632.77
Recursos Humanos	15	4200	630
Hardware	15	12.52	1.878
Software	15	5.98	0.897
Costes Indirectos	15	1054.625	158.19
Total	-	**5273.125**	**790.96**

Cuadro 3.7: Costes contingencia.

- **Avería dispositivo Hardware.** En caso de avería de algún dispositivo hardware sería necesario la compra de uno nuevo. Esto es poco probable, y es por eso que asignamos un 5 % de probabilidad.
- **Retraso en el la realización de algunas tareas.** Viendo el desconocimiento de algunas tecnologías que forman este proyecto, es necesario contemplar un porcentaje de retraso de algunas tareas. En este caso se harían unas 5 o 10 horas más a la semana, haciendo que sean en total unas 50 horas de retraso con una probabilidad del 30 %.

En la siguiente tabla se muestra el coste total de imprevistos.

Tipo	**Probabilidad (%)**	**unidades**	**Coste (€)**
Dispositivos Hardware	5	4	90
Retrasos	30	50 (horas)	120
Total	-	-	**210**

Cuadro 3.8: Costes de imprevistos.

Coste total.

Por ultimo se muestra el coste total del proyecto. Se calcula como la suma de costes directos, costes indirectos, contingencias e imprevistos.

3.4.2. Control de gestión.

La desviación más probable del proyecto es el retraso en la realización de algunas de las tareas descritas en el apartado de *planificación temporal*. El aumento de las horas de trabajo es algo que se tiene que controlar, ya que su desorganización podría encarecer aún más el presupuesto total, por el simple echo de ser un proyecto donde la mayor parte del presupuesto está dedicado a los costes de recursos humanos. Es por eso que, en los costes de los imprevistos

Tipo	**Coste (€)**
Costes directos	4218.5
Costes indirectos	1054.62
Contingencias	790.96
Imprevistos	210
Total	**6274.08**

Cuadro 3.9: Coste total.

es lo que más está destacado, ya que hace que el presupuesto aumente por más horas de trabajo.

Para llevar un control durante todo el desarrollo del proyecto, se utilizaran diferentes mecanismos, los cuales están comentados en el apartado de *metodología del proyecto*. Gracias a los mecanismo de seguimiento y de impulsar una metodología ágil, Scrum en este caso, la desviación del retraso de las tareas se va a controlar mucho más. En la reunión semanal, si hay alguna tarea que tarda más de lo esperado, se buscará la mejor solución para que afecte lo menos posible a las horas estipuladas de la realización del proyecto.

3.5. Desviaciones de la planificación inicial.

En este apartado se muestran los diferentes cambios de planificación que se han producido durante el desarrollo del proyecto. No han sido muchos, pero los tenemos en cuenta en este informe.

3.5.1. Dedicación y previsión temporal

Como se puede ver en la tabla, el número total de horas han aumentado en 5. Esto ha sido provocado por las numerosas experimentaciones y test de errores que se han hecho al código inicial del proyecto. Probar cada una de las soluciones para solventar los errores te hacia perder un gran tiempo de ejecución, ya que debes salir a la calle para probarlo, aumentando así las horas de trabajo.

Pero no todo es negativo, gracias al aumento de horas dedicadas tanto a la implementación y a la ejecución de los diferentes experimentos, ha provocado que las horas destinadas al desarrollo del algoritmo de rastreo haya disminuido, por el conocimiento obtenido de la nueva tecnología geofence y de su comportamiento en background. Además, los números experimentos durante la corrección de errores ha provocado también una disminución en las horas dedicadas a la tarea de análisis de datos, ya que cada vez que volvías de una ruta se tenía que analizarlos para ver si eran correctos o no.

En resumen, decir que la dedicación total en horas del proyecto no ha sufrido

Tarea	**Inicial (Horas)**	**Final (Horas)**
Búsqueda y estudio del modelo	20	20
Gestión del proyecto	75	75
Familiarización, análisis general	35	35
Diseño e implementación	55	80
Experimentación y corrección de errores	110	140
Análisis de datos	100	70
Algoritmo de rastreo basado en Geofence	50	30
Evaluación de rendimiento	20	20
Memoria final y defensa	60	60
Total	**525**	**530**

Cuadro 3.10: Dedicación en horas por cada tarea.

muchos cambios, solamente se ha sufrido un aumentado de 5 horas del total.

Diagrama de GANTT definitivo y dependencias entre tareas

En este apartado se muestra el diagrama de GANTT definitivo con todas las modificaciones del apartado anterior. Cabe destacar que la fecha de inicio y final de cada una de las acciones han sido modificadas pocos días para poder acabar el proyecto a finales de mayo, para así proceder a documentar. Además, se muestran todas las dependencias que se encuentran entre tareas.

3.5.2. Gestión económica

El aumento de 5 horas de trabajo en comparación a la planificación inicial no afecta al presupuesto inicial, ya que al hacer el calculo de los costes en imprevistos ya se contempló la posibilidad de aumento de horas de trabajo por contratiempos, con un total de 50 horas. Esto es gracias al buen control de gestión que se concretó inicialmente.

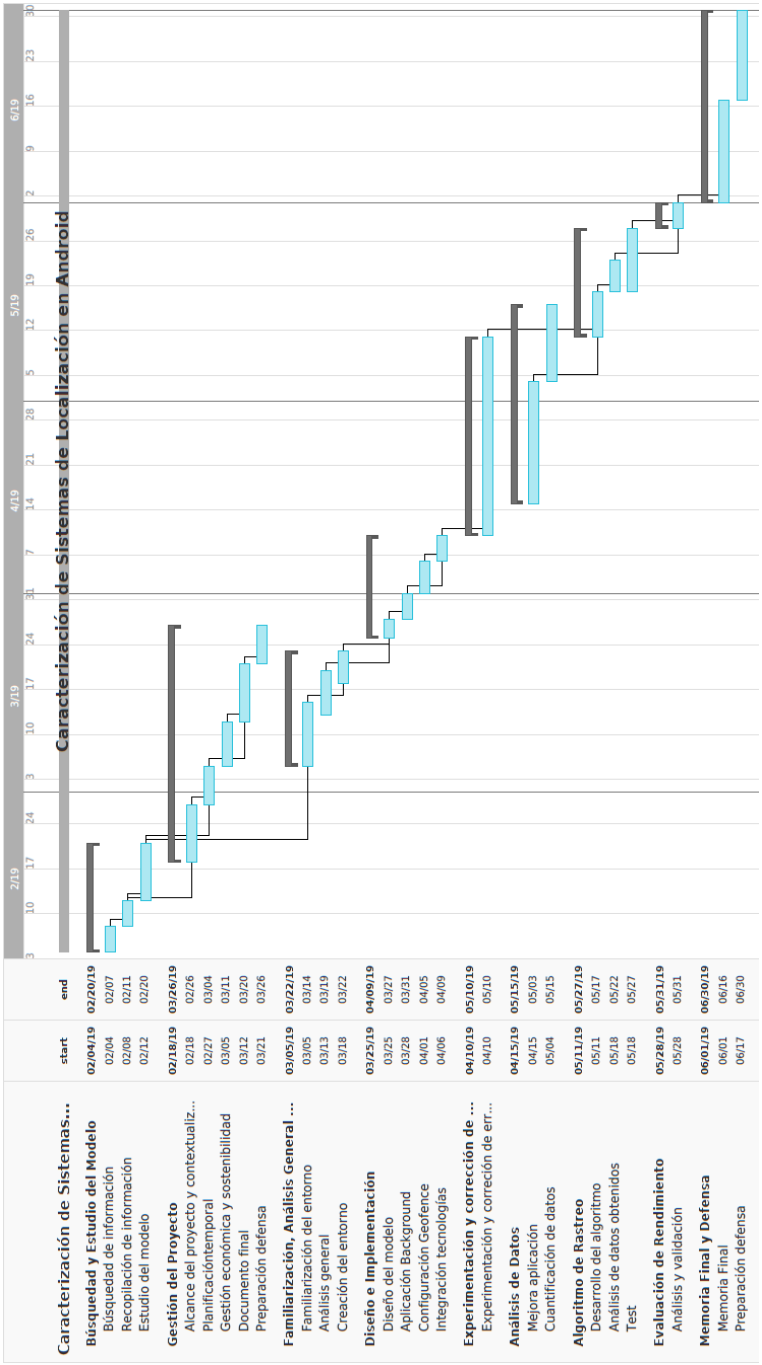


Figura 3.2: Diagrama de GANTT definitivo

Capítulo 4

Sostenibilidad y compromiso social

A continuación se resume el impacto del proyecto en la matriz de sostenibilidad y posteriormente se describen detalladamente.

	PPP	Vida Útil	Riesgos
Ambiental	Consumo del diseño	Huella ecológica	Riesgos ambientales
	4/10	14/20	0/0
Económico	Factura	Plan de viabilidad	Riesgos económicos
	8/10	14/20	-5/0
Social	Impacto personal	Impacto social	Riesgos sociales
	8/10	17/20	-2/0
Rango sostenibilidad	21/30	45/60	
	59/60		

Figura 4.1: Matriz de sostenibilidad.

4.1. Dimensión ambiental.

Para poder estimar el impacto ambiental del proyecto se han identificado cuáles son los recursos que requieran de un consumo de energía, estimado en kWh.

Teniendo en cuenta que un ordenador consume en media 180W[1] y que un teléfono móvil consume durante su carga unos 5W[2]; podemos estimar cuánto han consumido durante la realización de este proyecto. Las horas estipuladas en su utilización viene dado del apartado *gestión económica*.

[1] https://hardzone.es/2015/03/31/cuanto-cuesta-la-electricidad-que-consume-tu-pc/
[2] https://www.xataka.com/moviles/no-imaginas-el-dinero-que-te-cuesta-cargar-el-smartphone

- **Consumo de los ordenadores.** 550 h x 180 W = 99kWh.
- **Consumo de carga de los teléfonos móviles.** 78 h x 5 W = 1890Wh.
- **Consumo de la luz e Internet.** 525 h x 0.24kWh[3]. W = 126KWh.

El consumo total del proyecto ha sido de unos 226.90 KWh. Para disminuir este impacto ambiental, se ha evitado utilizar dos ordenadores a la vez y se ha intentado rebajar el consumo de luz en la habitación de trabajo, haciendo que este proyecto contenga mejoras ambientales respecto a otros similares.

Por otro lado, el desarrollo de este proyecto podría tener un impacto ambiental bastante elevado si no se tiene en cuenta la reutilización de recursos, ya que para abordar el problema de este proyecto, es necesario utilizar dispositivos móviles para las diferentes pruebas de experimentación. Para eso se ha decidido reutilizar los recursos con la utilización de solamente dos dispositivos móviles para todos los entornos de experimentación, algo que en una empresa no se consideraría, ya que necesitarían probar con los diferentes móviles en el mercado para conocer si existen diferentes comportamientos.

Durante la realización del proyecto, existe un experimento el cual ha podido empeorar la huella ecológica al probar nuestra implementación software durante un trayecto en coche. Estos es algo que, si el proyecto se volvería a hacer, se podría plantear de hacerlo de otra manera o buscar otro tipo de experimento. Por ejemplo, en este caso, la utilización de una moto eléctrica durante ese experimento podría ser un escenario de aumento de la huella ecológica.

4.2. Dimensión económica.

La estimación del coste inicial del proyecto suma un total de 6274.08€. Para el calculo de estos costes, se ha tenido en cuenta el coste de los recursos humanos y materiales necesarios para el correcto desarrollo del proyecto.

Viendo los detalles en el apartado de *gestión económica*, se puede apreciar que la mayoría de costes viene dado por recursos humanos, imprevistos y por el nivel de contingencia. Es por esto que hemos decidido que el desarrollo del proyecto es económicamente viable, ya que si hubiera un elevado coste de los recursos materiales nos podríamos plantear la no realización de este proyecto. Además, cabe destacar que, decidir utilizar recursos software de código libre ha provocado una disminución considerable en el presupuesto del proyecto. Asimismo, como se ha comentado en el apartado anterior, la utilización de solamente dos teléfonos móviles en la fase de experimentos ha provocado una reducción algo importante también en la gestión económica.

Finalmente, como hemos comentado en el apartado de *Dimensión Ambiental*, para abordar el problema presentado en este proyecto, es necesario la utilización

[3]https://tarifaluzhora.es/info/calcular-consumo-electrico-casa

de un elevado número de teléfonos móviles para controlar todos los entornos de experimentación. La decisión de utilizar solamente dos teléfonos, ha provocado una disminución considerable en el coste de los recursos materiales, algo que mejora económicamente este proyecto en comparación con otros similares. Además, si este proyecto se desarrollara en una empresa el presupuesto sería completamente distinto, ya que no estaría solamente un trabajador desarrollando este proyecto, sino que trabajarían más personas, aparte de que su sueldo sería superior al de un becario.

4.3. Dimensión social.

Al ser un proyecto dedicado a la investigación, pocos son los aspectos sociales que ayuden a mejorar la condición de vida de las personas. Pero también, cabe decir que el desarrollo de este proyecto es una necesidad real, ya que es poca la gente que ha hecho público sus trabajos sobre el tema que este proyecto trata, y es por eso que pienso que este proyecto puede aportar un bien común para esas personas interesadas. Es decir, se proporcionará un conocimiento público de tecnologías de ubicación en Android que antes no se tenía, y así hacer que el colectivo interesado en este tema tenga la facilidad de obtener esa formación.

Socialmente, como he dicho antes, no es un proyecto que mejore la calidad de vida de las personas, pero si podría molestar en un futuro, ya que muchas son las empresas que podrían utilizar este proyecto para saber cómo obtener datos de ubicación y proceder a su análisis creando diferentes perfiles de carácter personal. Con esto quiero decir que, podría ser desagradable para algunas personas la recogida de sus datos de ubicación, ya que nadie quiere que terceras personas tengan el conocimiento de sus pasos, algo que podría dejarlos en una situación de debilidad. Pero esto, es algo que no recoge este proyecto, ya que su objetivo no es esa recogida, sino entender como funciona la tecnología de ubicación en un teléfono Android.

Por último tengo que comentar que, con la realización de este proyecto, adquiriré a nivel personal muchos aspectos positivos. Primeramente, mi nivel de conocimiento de tecnologías relacionadas con el ámbito que estudio crecerá, ya que conoceré nuevos aspectos que actualmente eran desconocidos para mí. Además, obtendré una experiencia que puede ser muy valorada en un futuro cercano. Además, creo que afrontando los diferentes retos que puedan surgir durante la realización de este proyecto me ayudará a crecer como persona. Seré expuesto a nuevas situaciones que hasta ahora no se me han presentado, y con su correcta realización, podría conocerme más a mí mismo.

Capítulo 5

Sistemas de localización en Android

5.1. Introducción

Como se ha dicho anteriormente, hoy en día no se encuentra información práctica que explique realmente el verdadero comportamiento de la tecnología Geofence y cómo es su correcta forma de programación en los dispositivos móviles actuales, con versión Android superior al 8. Por este motivo, la primera sección tratará sobre la caracterización de los sistemas de localización en Android y la razón por el cual ha surgido la necesidad de tener una tecnología llamada Geofence.

El contenido de este apartado es un poco repetitivo en algunos conceptos a los comentados en la sección de *Estado del arte*, pero esta vez se entrará más al detalle.

Para empezar, se definen los conceptos más relevantes que se utilizarán durante el desarrollo del proyecto. Para cada uno de los términos se expondrá su definición y su funcionalidad. En segundo lugar, veremos cómo es capaz un dispositivo móvil de obtener una ubicación. Posteriormente, se caracterizarán las diferentes tecnologías que te proporciona Android para obtener una localización. En este punto, es conveniente explicar la tecnología Geofence y cuáles son sus métodos de trabajo. Y para concluir, se hará un análisis de lo qué realmente explican los artículos de investigación sobre Geofence y cuál es su interés real.

5.2. Objetivos

Los objetivos de esta parte del proyecto son los siguientes:

- Dar un primer enfoque de los conceptos que se verán durante el proyecto.
- Conocer cómo es capaz un teléfono móvil de obtener una ubicación.

- Entender el funcionamiento de la tecnología Geofence.
- Investigar métodos de trabajo de Geofence.
- Analizar hasta qué punto explican el funcionamiento Geofence en diferentes artículos de investigación.
- A partir de toda la información presentada, elegir la mejor arquitectura software para encaminar los diferentes objetivos del proyecto.

5.3. Desarrollo

5.3.1. Terminología y conceptos

Antes de empezar a entrar en detalle, es conveniente que se definan algunos conceptos sobre la localización en Android.

Localización

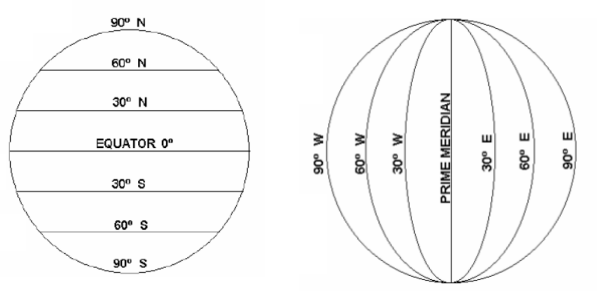
En este proyecto definimos localización como cualquier punto en el espacio representado en coordenadas geográficas, latitud y longitud. Estos dos conceptos se entienden como:

- Latitud: Se define como la distancia en grados desde el ecuador al norte y al sur. Por ejemplo, hay 90 grados de latitud desde el ecuador hasta cada uno de los polos, norte y sur.
- Longitud: Se define como la distancia en grados desde el primer meridiano al este o al oeste. Por ejemplo, todas las lineas de longitud pasan a través del polo norte y sur.

Así pues, cualquier punto en la superficie de la Tierra será representado por la intersección de la latitud y la longitud. Entonces, gracias a un proveedor de ubicación en nuestro dispositivo móvil que te suministra una latitud y longitud, puedes ser capaz de ubicarlo dentro de la superficie terrestre.

Interfaz de programación de aplicaciones

Interfaz de programación de aplicaciones, o generalmente llamado API, es una herramienta muy importante para el desarrollo de software en aplicaciones Android. En el apartado *Estado del arte*, se define como métodos que ofrecen diferentes librerías para ser utilizada por otro software como una capa de abstracción [3]



(a) (b)

Figura 5.1: Sistema de coordenadas geográficas. [Figura a] Latitud. Recuperado de [23]. [Figura b] Longitud. Recuperado de [23].

Una API es un conjunto de funciones que permite a los desarrolladores de software de Java crear programas simplificando, en gran medida, el trabajo de un realmente creador de software, ya que no debe programar las funciones desde cero. En conclusión, permite al programador usar funciones ya definidas para interactuar con un programa o sistema operativo [27].

Sirva de ejemplo: Google Maps API, Geofence API, GSON de Google, Android Pay, entre muchas.

Ejecución en primer plano

El servicio en primer plano, o dicho en inglés "foreground", es el estado que se encuentra una aplicación cuando el usuario puede interactuar con ella a través de una Activity o servicio. Es un claro ejemplo la aplicación de Google Maps cuando nos indica el camino a seguir.

Android considera que una aplicación se encuentra ejecutándose en primer plano si se cumple algunas de las siguientes condiciones [26]:

- Hay una actividad visible por el usuario. Esta actividad puede que se haya iniciado o en pausa.
- Tiene un servicio en primer plano.
- Otra aplicación en primer plano está conectada a la aplicación.

Ejecución en segundo plano

El servicio en segundo plano, o dicho en inglés background es el estado que se encuentra una aplicación cuando no es visible por el usuario y cuando no se ejecuta en el thread principal del teléfono. Exponiéndolo de otra manera, es un proceso que se encuentra en ejecución oculto para el usuario.

Su utilidad puede tener muchos fines, como por ejemplo sincronización con un servidor back-end, obtención de nuevo contenido de una aplicación, o incluso, la obtención periódica de ubicación, cómo queremos hacer en este proyecto.

Por contrapartida, Android define una aplicación ejecutada en segundo plano, si no se cumple ninguna condición que define el estado en primer plano [26].

Sistema de Posicionamiento Global

O dicho en inglés Global Positioning System (GPS) es un sistema mundial de navegación para proporcionar información de ubicación de cualquier punto en la Tierra. Se determina midiendo la distancia desde un grupo de satélites en el espacio.

Fused Location

Fused location es el principal proveedor de ubicación que encontramos en la plataforma Android hoy en día. Este sistema es capaz de aprovecharse de diferentes señales de los sensores para determinar la localización del dispositivo móvil. No es una tarea fácil, ya que su propósito es buscar una solución para encontrar una ubicación siempre pensando en la eficiencia de la batería.

El motivo por lo que surge Fused location, algo que ya he comentado en el apartado de *Estado del arte*, es la alta demanda de batería al utilizar GPS para obtener una localización del dispositivo. Entonces, Fused Location utiliza siempre la mejor forma de obtener una ubicación, ya sea por GPS, WiFi, red celular como la combinación de diferentes sensores integrados en el dispositivo.Google define este proveedor como la mejor solución de ubicación respetando siempre al impacto de la descarga de batería en la obtención de ubicación.

Android, proporciona una API para especificar la calidad de servicio que necesitas de localización. Fused location puede soportar estos escenarios[29]:

- Última localización conocida. Petición de la última localización conocida por el dispositivo móvil. Una buena estrategia para empezar la aplicación es la de pedir la ultima localización conocida.
- Actualizaciones de ubicación. Además de la última localización conocida, podemos obtener localizaciones de ubicación definiendo un intervalo de tiempo.
- Ajustes de ubicación. Al hacer peticiones de ubicación puedes definir la calidad de servicio deseado y el tipo de proveedor de las localizaciones provenientes. Algo que se detallará más adelante.

Geofence

Como se ha dicho en el apartado *Estado del arte*, Geofence se define como una región circular ubicada en cualquier punto del mapa, dada una longitud, latitud y radio; y te notifica cuando el dispositivo móvil entra, sale o permanece dentro de la región del Geofence durante un cierto tiempo.

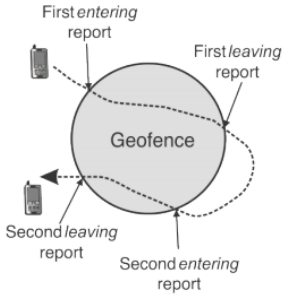


Figura 5.2: Eventos de transición en Geofence. Location events for triggered services. Recuperado de [17].

Un ejemplo muy típico de este sistema podría ser la de aplicación que defina una Geofence alrededor de un aeropuerto. Cuando se da el caso de que una persona se dirige a la puerta de embarque y el dispositivo cruza la Geofence, la aplicación envía una notificación para que el usuario vea la tarjeta de embarque.

Este sistema utiliza, de forma inteligente, los sensores integrados al dispositivo para detectar cuando un dispositivo cruza una Geofence respetando el impacto de la bateria [30].

Para todo esto, Google te proporciona una API para que puedas utilizar este sistema de forma más intuitiva y fácil.

Push Notification

Push notification o Push technology es un mecanismo que te permite enviar diferentes mensajes a los usuarios de una aplicación y se muestra en la barra del notificaciones en el teléfono móvil.

Durante el desarrollo de este proyecto, se utiliza este mecanismo para empezar el rastreo basado de Geofence. Para enviar el mensaje Push Notification utilizamos un sistema que nos proporciona Firebase [1] llamado Firebase Cloud Messaging (FCM) [2].

[1] https://firebase.google.com/?hl=es-419
[2] https://firebase.google.com/docs/cloud-messaging

5.3.2. Capacidad del dispositivo móvil de obtener una ubicación

La ubicación en teléfonos móviles inteligentes se pueden obtener de cuatro formas diferentes:

Proveedor de GPS en Android

¿Cómo el teléfono móvil obtiene una ubicación por GPS?

El proveedor de GPS en Android utiliza satélites para determinar la ubicación actual del dispositivo móvil. Este sistema es de los que mayor precisión tienen, pero encontramos un inconveniente, la cantidad de batería que gasta.

Todo el sistema de GPS consiste en una red de 24 satélites donde almenos 5 están a la vista desde todos los puntos del planeta. Además, para controlar estos satélites, hay presente 5 estaciones de monitorización y 4 antenas ubicadas en todo el mundo, con el fin de recolectar los datos de ubicación de los satélites y proporcionar una retransmisión de datos de corrección en caso de fallo [31].

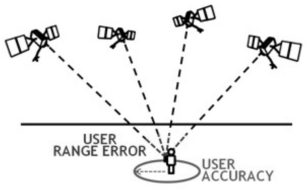


Figura 5.3: Sistema GPS. Recuperado de [24]

Para el cálculo de la ubicación del dispositivo, este sistema utiliza un método llamado triangulación, que consiste en:

Las señales retransmitidas por cada satélite, son utilizadas por el receptor para calcular su posición, estimando el tiempo de viaje de la señal para calcular la distancia que le separa con el satélite. Para efectuar este cálculo correctamente, el dispositivo móvil debe de obtener señales GPS de al menos tres satélites [31].

Como se puede ver en la Figura 5.4, es posible deducir la posición a partir de la intersección de las tres áreas de los satélites, obteniendo latitud, longitud y altitud del usuario. Además, siempre y cuando tengamos cuatro o más satélites a la vista, obtendremos más precisión en la ubicación. Por otro lado, a partir de 4 satélites, podemos obtener otro tipo de datos de ubicación a partir del cálculo 3D de la posición de usuario, como la latitud, la velocidad, la trayectoria, etc.

Con este método se consigue una precisión GPS que va de 18 metros a 100 metros, muy adecuado para todo tipo de aplicación [31].

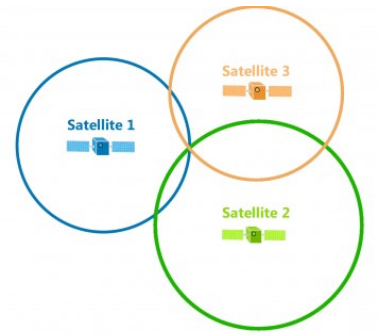


Figura 5.4: Método triangulación GPS. Recuperado de [28]

Diferentes escenarios

La presencia del chip GPS origina la distinción de dos tipos de localización en referencia al lugar dónde te ubiques, localización en exteriores e interiores. Esta diferencia se produce por la baja precisión del GPS cuando trabaja en ambientes interiores, ya que la comunicación entre el satélite proveedor de ubicación y tu teléfono móvil no tienen una comunicación directa.

- Localización en exteriores. Se produce cuando el dispositivo móvil sostiene una comunicación directa con los diferentes satélites proveedores de ubicación. El margen de error en este contexto es como máximo de 5 metros [24].
- Localización en interiores. Se produce cuando el dispositivo móvil se encuentra sin una comunicación directa con los diferentes satélites proveedores de ubicación. Por ejemplo, este fenómeno se suele producir cuando estás dentro de un edificio. Para solucionar la obtención de ubicación en interior se utiliza WiFi llamado 802.11mc, que calcula el tiempo que tarda la señal en viajar desde nuestro móvil al router [25].

Al ser un proyecto dónde todos los experimentos están orientados en rastrear los pasos de una persona, es conveniente decir que este proyecto está orientado en localizaciones de exteriores.

Limitaciones

Para este sistema podemos encontrar las siguientes limitaciones:

- Rápida descarga de batería cuando utilizas este sistema en tiempo real.
- El vapor de agua u otras partículas de la atmósfera puede retrasar la señal de propagación.

- Multipath Fading. Se produce cuando una señal rebota con objetos provocando un retrasa en su propagación.
- Las señales GPS no llegan a ambientes interiores.

WiFi

El servicio de ubicación WiFi es la solución de obtener localización en ambientes interiores, ya que el GPS no funciona. Al ser un rango más corto la localización por WiFi es significativamente más precisa, y permite una buen rastreo en interiores. De esta forma, podemos tener una ubicación muy precisa sin la necesidad de utilizar el sistema GPS.

¿Cómo el teléfono móvil obtiene una ubicación por WiFi?

El proceso de obtención de ubicación por WiFi es muy similar al sistema que se ha explicado del GPS, pero esta vez, la distancia que separa el dispotivo móvil al punto de acceso es mucho más pequeña.

En este caso, se utiliza un estándar llamado WiFi Alliance, basado en el cálculo del tiempo que tarda un dato en retransmitirse. Para conseguir esto, se utiliza el método FTM (Fine Timing Measurement) especificado por el estándar IEEE 802.11mc, que es capaz de calcular la distancia y la posición exacta en la que se encuentra un dispositivo móvil de un punto de acceso WiFi [34].

Este método también utiliza la característica de la triangulación explicado en GPS, ya que obtiene una ubicación a partir del cálculo de la intersección de 3 áreas de cobertura de diferentes puntos de acceso, es decir, necesitamos también la presencia de más de 3 puntos de acceso para determinar una localización. Asimismo, si añades más puntos de acceso, puedes obtener un escenario 3D para calcular mucha más información. Parecido al GPS, pero con distancias mucho más pequeñas.

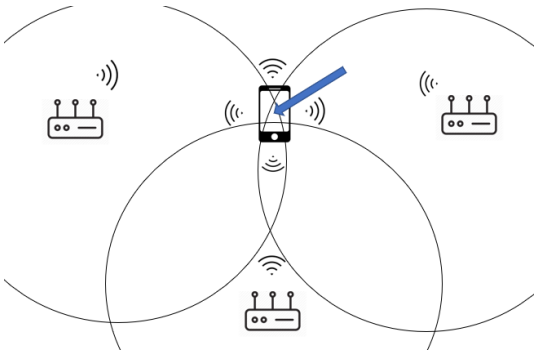


Figura 5.5: Método triangulación WiFi. Fuente: elaboración propia.

Con el requerimiento de tener una posición exacta, este sistema debe sincronizar los relojes. Para conseguir esto, el punto de acceso marca su paquete con la hora de envío, y cuando llega al dispositivo, éste puede calcular la distancia que les separa (Distancia = Velocidad Luz x tiempo). Este procedimiento efectuado una vez no funciona, ya que muchos factores pueden alterar la distancia de propagación, y es por eso que este calculo se repite muchas veces por segundo para mejorar su precisión [34].

Limitaciones

En Wifi dependemos de varios factores:

- Ancho de banda del canal. Si hay una red muy colapsada y con mucho trafico pueden haber muchas retransmisiones y interferencias. Cuanto más ancho de banda, mayor será la precisión
- Multipah Fading. El mismo problema que con el GPS, los rebotes de la señal hacen posible que recibas la señal demasiado tarde.

Proveedor de red celular

Es el método que tiene menor efecto en la batería, utilizando la ubicación de las torres de telefonía cercanas con un resultado pobre en precisión.

¿Cómo el teléfono móvil obtiene una ubicación por red celular?

Este sistema utiliza el mismo método de triangulación que hemos visto en el sistema GPS y WiFi, pero obteniendo una precisión que va de los 500 metros a 1500 metros. En zonas urbanas el margen de error es menor, ya que hay un número mayor de torres celulares.

Limitaciones

El problema que encontramos en este tipo de sistema de localización es su alto margen de error. Esto es debido a muchos factores:

- Zonas rurales.
  - El bajo número de torres celulares en la zona, hace que en zonas rurales la precisión de ubicación con este método sea muy poco preciso o incluso nulo.
- Zonas urbanas.
  - Multipath Fading. La existencia de numerosos edificios y obstáculos que hacen rebotar las ondas de las torres celulares hacen que la

precisión baje. En contrapartida, el incremento de torres celulares en estas zonas mejora bastante la precisión gracias al método de la triangulación.

Proveedor de ubicación pasivo

Es un tipo especial de proveedor de ubicación, ya que no hace falta activarlo. Con esto quiero decir que, obtiene una ubicación a partir de peticiones que otras aplicaciones hacen al dispositivo móvil. En este caso, el consumo de batería es insignificante, ya que te aprovechas de las peticiones que hacen otras aplicaciones, pero por otro lado, dependes totalmente de ellas.

Sensores teléfono móvil

¿Cómo el teléfono móvil obtiene una ubicación a partir de los sensores ubicados en el teléfono móvil?

Los teléfonos actuales tienen incorporados diferentes tipos de sensores que pueden medir el movimiento, la orientación y diversas condiciones ambientales. Cabe decir que, esta explicación se va a centrar en sos sensores que miden el movimiento y la posición, los cuales se corresponden al acelerómetro, giroscopio, magnetómetro y giroscopio.

Figura 5.6: Ejemplo sensores dispositivo móvil. Recuperado de [28]

Los sensores son capaces de proporcionarnos datos en crudo, en otras palabras, datos que no han recibido ningún tratamiento, con el objetivo de que se trabajen y se saque un provecho individual de cada uno de ellos. Pero existe la situación de relacionar, en tiempo real, los datos de diferentes sensores a la vez (fusión de sensores), para hacer cosas que con uno solo no podríamos hacer, como por ejemplo, saber tu movimiento,los pasos que has hecho o incluso predecir una futura localización [46].

Para trabajar la ubicación a través de los sensores del teléfono móvil, se debe utilizar conjuntamente con algún proveedor mencionado anteriormente y es por eso que surge la tecnología Fused Location, proporcionado por Google como una API de proveedor de ubicación. Fused location utiliza múltiples fuentes,

incluidos sensores del teléfono, para proporcionar la mejor solución de obtener una localización respetando el consumo de batería del dispositivo.

5.3.3. Ejecución en segundo plano en Android

Antes de explicar en profundidad la tecnología Geofence, es importante hablar sobre las limitaciones de ejecución en segundo plano en dispositivos Android y como afecta en la obtención de una ubicación. Cabe destacar que esta sección es todo información teórica de paginas oficiales de Android y de vídeos proporcionados por Google, los cuales no muestra resultados en la práctica.

Tareas en segundo plano

La plataforma Android soporta 2 maneras diferentes de ejecutar en segundo plano: [45]

- Threads. Este sistema realiza un procesamiento asíncrono utilizando las classes ThreadPools y Executor. Este método necista la sincronización de los diferentes Threads en ejecución, utilizando las clases android.os.Handler o AsyncTasks
- Servicio. Permite operaciones de larga ejecución en segundo plano sin interfaz de usuario. Este es el método que se utilizará durante el desarrollo de este proyecto. Lo bueno de esto es que aunque este ejecutándose otra aplicación, el servicio en segundo plano sigue ejecutándose.

Proceso de elección de la mejor arquitectura

En la versión Android 5.0, se introdujo una nueva API, JobScheduler, con el fin de programar servicios o tareas para ejecutarse en segundo plano.

Google te recomienda utilizar la arquitectura software que presenta como WorkManager[3]. Para la elección de esta arquitectura, google te proporciona un diagrama con las siguientes preguntas:

[3]https://developer.android.com/topic/libraries/architecture/workmanager

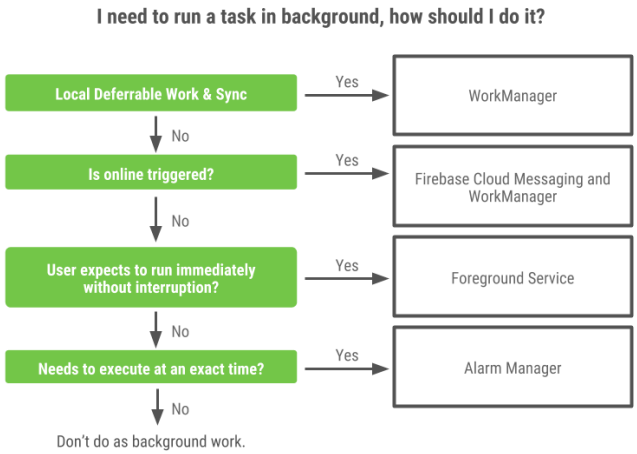


Figura 5.7: diagrama ejecución en segundo plano. Recuperado de [35]

Viendo esto, nosotros debemos escoger la segunda opción de Work Manager y Firebase Cloud Messaging. También se podría hacer como Foreground Service, pero no es un requerimiento de este proyecto.

Android define Work Manager como[35]:

> WorkManager es una biblioteca de Android que ejecuta en segundo plano procesos diferentes cuando se cumplen restricciones del trabajo.
> WorkManager está destinado a tareas que requieren una garantía de que el sistema las ejecutará incluso si se cierra la app.

Gracias a toda la información recogida, se ha decido escoger la API JobScheduler como la herramienta principal en la ejecución de segundo plano, trabajando conjuntamente con Geofence y Fused Location. Cabe destacar que WorkManager se encuentra ahora mismo en fase beta[35], y es por eso que nosotros utilizaremos directamente la API JobScheduler, ya que es lo que WorkManager utiliza internamente.

Limitaciones en tareas segundo plano

"*Location in background is power hungry*".[36]

Las aplicaciones en segundo plano son las mayores contribuidoras de problemas en la vida de la batería. Es muy agresivo hacer peticiones de localización en background y está muy limitado su utilización en Android

- ¿Porque la existencia de limites en segundo plano? [36]

Las limitaciones de ejecución en segundo plano en las versiones 8 o superior, aparecen con el objetivo principal de conseguir una vida de batería de varios días.

- Ahorro de batería.
  - Con las técnicas que comentaremos más adelante, Android es capaz de aumentar su eficiencia de la batería de 6 hasta 8 veces mas.
  - Reduce el impacto de batería en los servicios de localización en segundo plano y de escaneos de WiFi.
- Gestión de la memoria RAM.
  - Evitar mensajes de Broadcast para despertar aplicaciones.
  - Limita aplicaciones ejecutándose en servicios de Background.
- Prepara un escenario para el futuro.
  - Se quiere que en un futuro cercano, tanto las tareas como las alarmas, gestionen mucho mejor los recursos de los dispositivos.

- Evolución restricciones en segundo plano.

"*Batería mas inteligente y eficiente*"[4]

Esta frase apareció en mayo de 2015 como el principal eslogan del lanzamiento del Android 6.0 Marshmallow, primera versión en Android en implementar restricciones para ejecuciones en segundo plano. Con el lanzamiento de esta versión, se empezaron a ver los primeros focos a la importancia de encontrar soluciones al impacto de batería de algunas aplicaciones.

Esta versión de Android introduce el concepto Deep Doze. Consiste en poner el dispositivo en un sueño profundo cuando éste no detecta ningún tipo de movimiento. Por ejemplo cuando se encuentra en una mesa. Básicamente, este sistema apaga toda activa en ejecución en segundo plano con el objetivo de extender la vida de batería.

Con la siguiente versión de Android, Android 7.0 Nougatm aparece lo llamado Light Doze. Es el mismo funcionamiento de Deep Doze, pero admite la condición de que el móvil pueda estar en movimiento. Este sistema no es tan restrictivo eliminando todas los procesos en background, sino que elimina los que tienen menos prioridad.

[4] https://www.android.com/intl/es_es/versions/marshmallow-6-0/

Más tarde, con la aparación de Android 8.0 Oreo, surgen las restricciones más severas en cuanto localización en background. Aparece lo llamado restricciones de batería adaptativo [37]. Consiste en:

- Grupos de aplicaciones donde se ordenan en aplicaciones activas, working set, frecuentes y raras. Donde activas las que tienen más prioridad y las raras las que menos. Decir que FCM significa Firebase Cloud Messaging.

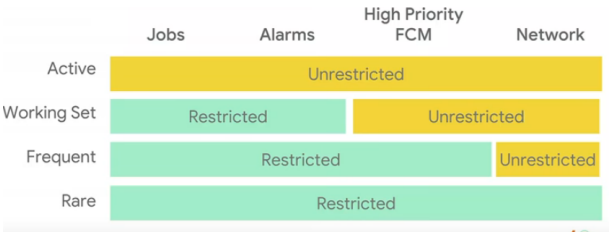


Figura 5.8: Restricciones por grupo en aplicaciones. Recuperado de [37]

- Presencia de un algoritmo adaptativo, basado en Maching Learning, el cual estudia diferentes patrones de comportamiento de los usuarios y si ve que una aplicación se va a utilizar dentro de unas 2 horas las pone activas[37]. Si una aplicación no se utiliza durante un tiempo, las va colocando en grupos con menor prioridad. Esto es tan potente que es capaz de percibir con una cierta probabilidad qué aplicación vas a abrir según la hora y día de la semana que es.

Todo lo mencionado arriba, es preocupación solamente del sistema operativo y no de los desarrolladores de software.

Restricciones hasta ahora:

- Tareas, alarmas, servicios
- Internet y FCM
- Actualizaciones de ubicación, aunque la pantalla esté apagada.

Estas restricciones también son aplicables cuando el dispositivo está cargando para respetar la privacidad del usuario.

En la nueva versión de Android, Android 9 Pie, cuando se pide la autorización de ubicación del teléfono, al usuario no se le pregunta sobre permitir o no el acceso, sino que ahora tiene 3 posibilidades: Que la aplicación pueda tener la localización siempre, que solo la pueda tener cuando se este utilizando la aplicación en primer plano o nunca.

- ¿Qué recomienda Google para aplicaciones de ubicación en segundo plano?

Con el objetivo de reducir el consumo de batería, con la aparición de Android 8.0, se limitan las actualizaciones periódicas de ubicación en aplicaciones en segundo plano, sin importar la versión SDK de la aplicación.

Dependiendo de la funcionalidad de la aplicación, puede ser que los resultados no sean los esperados cuando quieres obtener localizaciones periódicas, entonces Google te recomienda lo siguiente:[38]

- Pasar directamente la aplicación en primer plano
- Utilizar el servicio en primer plano. Se define como un servicio activo, donde siempre se muestra una notificación en la barra de notificaciones del teléfono conforme se esta ejecutándose la aplicación. Un claro ejemplo es cuando se cierra una aplicación de música pero se sigue escuchando, mostrándolo en la barra de notificaciones.
- Utilizar Geofence API, que optimiza el consumo de batería. Esto es lo escogido para este proyecto.
- Utilizar proveedor de ubicación pasivo.

5.3.4. Fused Location y Geofence

Geofence surge como una de las soluciones que te da Google para la utilización de peticiones de ubicación en aplicaciones ejecutandose en background. Para su correcto funcionamieno, esta tecnología debe estar construida junto a la instancia de Fused Location.

Razones para utilizar Geofence

El sistema de Geofence es un servicio que realiza un seguimiento de las coordenadas de dispositivos móviles y los compara continuamente con las Geofence, que son áreas circulares definidas por el usuario. Cuando un objeto móvil ingresa, está un cierto tiempo dentro o sale de un Geofence, el sistema emite una notificación

Geofence es la solución proporcionada por Google para ayudar a saltarse las limitaciones de ubicación en segundo plano en teléfonos móviles Android con una versión superior a la 8.

La principal razón de su utilización es su eficiencia. Su funcionamiento consiste en comprobar si el dispositivo móvil se encuentra fuera o dentro de la área definida, y si lo cree conveniente lanza una notificación. Esto es algo que se está

implementado y optimizado en el sistema operativo, y reduce significativamente la descarga de batería en comparación a actualizaciones de ubicación.

Técnicas para la utilización de Geofence

Durante el desarrollo de esta sección, se han pensado en 4 técnicas para la utilización de Geofence en una aplicación:

- Notificación Geofence. Recibir alertas al móvil si éste ha entrado o salido de un limite fijado como una Geofence.
- Proximidad a un punto de interés. Detectar la proximidad de un dispositivo móvil en relación a un punto de interés. Define el punto de interés en el centro de la Geofence, y cuando salta la notificación de entrada sabes la distancia que falta para llegar.
- Rastreo del dispositivo. Monitorizar un objeto móvil a través de su ruta a partir de Geofences dinámicas.
- Ruta preasignada. A partir de una ruta ya creada, seguir la progresión de un objeto en dicha ruta. Se podrían poner diferentes tiempos de llegada en cada Geofence y recibir una alerta si no se cumplen.

¿Qué cuenta Google sobre este servicio?

Los información que se expondrá a continuación viene dada por Android como las mejores prácticas en la utilización de Geofence. Adjunto el link a continuación: `https://developer.android.com/training/location/geofencing.html#HandleGeofenceTransitions`.

Radio óptimo de la Geofence

Android te recomienda que, para obtener los mejores resultados, el radio mínimo del Geofence se debe establecer entre los 100 y 150 metros. Hay casos, por ejemplo en zonas rurales, en que la cobertura de la red o la señal GPS no está disponible plenamente, entonces, el rango de precisión de ubicación puede variar de cientos de metros a varios kilómetros. En estas situaciones se debe aumentar el radio de Geofence a más grande.

Ahorro de Batería

- Usar radios de Geofence más grandes para las ubicaciones donde el usuario pasa una cantidad significativa de tiempo. En este caso, el teléfono móvil

consume menos ya que se reduce la frecuencia con la que la aplicación verifica la entrada o salida del dispositivo móvil de la Geofence.

- Establecer la capacidad de respuesta de notificación, en inglés Notification Responsiveness, a un valor más alto. Es importante decir que al hacerlo se mejora el consumo de energía al aumentar la latencia de las alertas de Geofence.

Problemas de latencia en avisos

El servicio de Geofence no consulta continuamente la ubicación. En general, la latencia es de menos de 2 minutos, incluso menos cuando el dispositivo se ha estado moviendo. Si los límites de ubicación de fondo están en efecto, la latencia es de aproximadamente 2-3 minutos en promedio. Si el dispositivo ha estado parado durante un período de tiempo significativo, la latencia puede aumentar (hasta 6 minutos).

Utilizar la transición de DWELL para reducir el SPAM

Utilizar DWELL en lugar de la transición de ENTER para determinar que un usuario realmente ha entrado en una zona. Esto es posible si en configuración de Geofence pones un tiempo de espera bastante pequeño, ya que para Geofence con radio de 100 metros las puedes cruzar en menos de 2 minutos.

¿Cómo es capaz de saber cuándo entra?

El sistema calcula la relación de superposición entre el círculo de localización y el círculo de la Geofence. Solo genera la alerta cuando esa relación es como mínimo de 85 % para una Geofence grande y un 75 % para una pequeña. Para el EXIT la relación es 15 % o 25 %. Fuera de estos marcos, el servicio de Geofence marca su estado como INSIDE_LOW_CONFIDENCE o OUTSIDE_LOW_CONFIDENCE y no se envía ninguna alerta.

Parámetros de configuración en Android

Para la integración a la aplicación a desarrollar de los diferentes servicios de ubicación, es necesario conocer todos los parámetros de configuración, ya que marcará el comportamiento de la puesta en marcha de cada servicio.

- Geofencing API

Primero empezamos con la creación de los objetos Geofences. Para ello se utiliza la clase Geofence-Builder para establecer el radio, la duración del Geofences y los tipos de transición. Una vez creados los objetos Geofence, se añade a una lista para proceder a la siguiente fase.

```
geofenceList.add(new Geofence.Builder()
    .setRequestId(geofence.id)

    .setCircularRegion(geodence.latitude, geodence.longitude, geofence.ratius
        )
    // duracion del Geofence el cual se define. Poner Geofence.NEVER\_EXPIRE
        si siempre esta presente
    .setExpirationDuration(geofence.expiration)
    // transiciones a avisar
    .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
        Geofence.GEOFENCE_TRANSITION_EXIT))
```

Segundo, hay que especificar que Geofences deben añadirse al sistema y cuándo van a lanzar una notificación. Para ello se adjunta a continuación un fragmento de código con todos los parámetros de configuración.

```
private GeofencingRequest getGeofencingRequest() {
    GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
    builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
    builder.addGeofences(geofenceList);
    return builder.build();
}
```

En este ejemplo, solamente se lanza una notificación cuando el sistema Geofence detecte una entrada en cualquier Geofence definida en geofenceList. Pero si se da el caso de que antes de esa entrada se ha efectuado una salida o espera, estas notificaciones también serán enviadas al teléfono juntamente con la entrada inicial. Esto pasa porque nosotros hemos definido estas Geofences con todas las tipos de transacciones, pero el aviso a la aplicación solamente lo hemos hecho con la entrada. A continuación se muestra esta explicación en modo de ejemplo:

- T = 1. Obtenemos la notificación al evento de transición de entrada en la Geofence 1.
- T = 2. Geofence no se despierta ya que no hay evento de entrada de ninguna Geofence.
- T = 3. Geofence se despierta con la entrada de Geofence 2, además obtiene la salida de Geofence 1

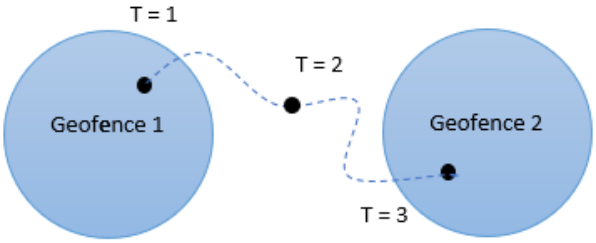


Figura 5.9: Ejemplo de eventos de transición. Fuente: elaboración propia

Podemos encontrar toda este información en: `https://developer.android.com/training/location/geofencing.html`

Fused Location Provider API

Esta parte es la más importante de entender para cada uno de los parámetros de configuración, ya que es lo que Android tiene restringido para versiones superiores o iguales a la 8.0.

- Intervalo de actualización de ubicación:

LocationRequest.setInterval(x), establece la velocidad en milisegundos en el que la aplicación recibe una actualización de ubicación. Android te recomienda poner este intervalo solo pocas por hora en aplicaciones que se ejecutan en segundo plano, ya que sino ese servicio va a ser restringido por el sistema operativo [41].

- Intervalo de actualización más rápido

setFastestInterval(x): Este método establece la velocidad más rápida en milisegundos en el que la aplicación puede obtener actualizaciones de ubicación. Google recomienda unos 5 minutos [41] para aplicaciones en segundo plano.

- Prioridad

setPriority(Name) establece la prioridad de la solicitud de ubicación según como queramos la exactitud de la ubicación. Cuanto más prioridad, más batería se gastará. A continuación se muestra todos los tipos de prioridad de ubicación:

- PRIORITY_BALANCED_POWER_ACCURACY. Es la recomendada por Google para ubicación en segundo plano y tiene una precisión de aproximadamente 100 metros. Con esta configuración es probable que la ubicación se obtenga a partir de WiFi o de torre celular para no gastar batería.
- PRIORITY_HIGH_ACCURACY. Esta configuración sirve para solicitar la ubicación más precisa posible. Con esta configuración es mas probable que la

localización se obtenga a partir de GPS.

- PRIORITY_LOW_POWER. Se utiliza para obtener ubicación consumiendo muy poca batería. Su precisión es de unos 10 kilómetros.
- PRIORITY_NO_POWER. Se utiliza cuando quieres recibir actualizaciones de ubicación cuando estén disponibles, sin que las pidas tu. Con esta configuración se recibe ubicaciones activas de otras aplicaciones.

A continuación se muestra un fragmento de código de la configuración de fused Location API:

```
1 fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);
2
3 LocationRequest locationRequest = LocationRequest.create();
4 locationRequest.setInterval(10000);
5 locationRequest.setFastestInterval(5000);
6 locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
7
8 fusedLocationClient.requestLocationUpdates(locationRequest, locationCallback,
      null /* Looper */);
```

Podemos encontrar toda este información en `https://developer.android.com/training/location/change-location-settings.html`

5.3.5. Análisis de estudios sobre Geofence

Como se ha comentado anteriormente, este proyecto será objeto de extracción de información para poder hacer diversos artículos de investigación sobre Geofence en Android. Es por eso que, después de caracterizar los diferentes sistemas de localización, se requiere definir un punto en este proyecto, dónde se estudie hasta qué punto se explica el comportamiento de Geofence desde el punto de vista de la investigación.

Para ello, se han seleccionado diferentes artículos de investigación basados en la tecnología Geofence, gracias a la herramienta proporcionada por la Universitat Politécnica de Catalunya, eBIB, un recurso que da acceso a diferentes revistas y libros suscritos por las bibliotecas de la UPC.

Después de una búsqueda profunda en la página web `https://ieeexplore.ieee.org/Xplore/home.jsp`, no se ha encontrado ninguna revista que hable directamente sobre la API de Geofence proporcionada por Google. La mayoría de revistas, exponen diferentes aplicaciones de Geofence basadas en informaciones solamente teóricas de otras revistas o de recursos públicos, como por ejemplo páginas web. Por otra parte, también hay casos que se obtienen resultados de Geofence solamente para demostrar que una aplicación funciona realmente bien, siempre bajo una implementación propia de Geofence, donde todas las áreas son definidas de forma rectangular o irregular.

En conclusión, se podría decir que la mayoría de artículos de investigación están orientados a un fundamento muy teórico del comportamiento de Geofence, pero no tienen en cuenta que en el mundo real la teoría puede cambiar por diferentes factores externos no controlados. Y es por eso que este proyecto es necesario para poder obtener información práctica del comportamiento de Geofence, y así poder elaborar artículos que puedan estar basados en información real de su comportamiento.

Capítulo 6

Software monitorizado basado en Geofence

6.1. Introducción

Después de tener claro cuál es la información pública proporcionada sobre la tecnología Geofence y cuál es su comportamiento teórico cuando se implementa como un servicio en segundo plano, llegamos a la sección de la implementación de esta tecnología.

Por lo tanto, se decide implementar un software que sea capaz de instalarse en un dispositivo Android a partir de la plataforma Android Studio[1]. Dicha aplicación, debe tener implementada la tecnología de Geofence en un servicio ejecutándose en segundo plano, que vaya recogiendo las notificaciones de entrada, espera y salidas de las diferentes Geofences.

Cabe destacar que, en esta sección, solo se implementarán Geofences estáticas, ya que queremos ver cual es funcionamiento en la práctica en un recorrido ya definido previamente por el desarrollador.

Primeramente, se empezará con el diseño del modelo a implementar en esta sección en referencia a toda la información adquirida en la sección anterior. Después, dará comienzo la implementación de solamente una tarea en segundo plano que sea capaz de escribir en un fichero. Cuando se haya conseguido, se hará una aplicación donde llevaremos el servicio Geofence a una aplicación en en primer plano, y al ver su funcionamiento correcto, finalmente, se integrarán con la tarea en segundo plano.

[1] https://developer.android.com/studio

6.2. Objetivos

Los objetivos de esta sección son los siguientes:

- Diseño de la arquitectura software de la aplicación a implementar.
  - El diseño deber estar basado en la información explicada en *Sistemas de localización en android*.
- Implementación de un servicio ejecutándose en segundo plano.
  - Como requerimiento el servicio debe escribir en un fichero del teléfono móvil.
- Implementación de Geofence.
  - Software ejecutándose en primer plano donde se vea exactamente que se reciben las notificaciones de las entradas y salidas de las diferentes Geofences definidas.
- Integración del servicio Geofence con ejecución en segundo plano.
  - El software debe permitir el recibimiento de notificaciones de Geofence mientras se ejecuta segundo plano.

6.3. Recursos

6.3.1. Android Studio

Es un entorno de desarrollo integrado que actualmente es la herramienta oficial para crear aplicaciones Android. Al ser exclusivo a la programación de aplicaciones para dispositivos Android, proporciona a Google un mayor control sobre el proceso de producción [40]. Android Studio es multiplataforma permitiendo su instalación de forma sencilla tanto en Windows como en Linux o Mac.

Para ejecutar este entorno de desarrollo es necesario disponer de la biblioteca de desarrollo Android basado en Java, llamado Android SDK Platform[2], en concreto, durante el desarrollo de esta sección se ha utilizado Android 8.1 (API level 27).

Se puede encontrar más información en `https://developer.android.com/studio/intro?hl=es-419`.

[2]https://developer.android.com/studio/releases/platform-tools

Figura 6.1: Logo de Android Estudio. Recuperado de [39].

Este recurso se ha utilizado para desarrollar las implementaciones software de esta sección. Tengo que decir que, al principio del proyecto, tuve un problema con su instalación, ya que todos los requerimientos que pide tener instalados en el ordenador ocupan demasiado. Pero por otra parte, gracias a la facilidad de trabajo que te proporciona el entorno de Android Studio, te das cuenta el por qué no tiene prácticamente competencia.

6.3.2. AVD Manager

Android Virtual Device (AVD) permite emular un dispositivo Android en un entorno virtual pudiendo elegir entre diferentes resoluciones de pantalla y todas las versiones Android disponibles.

Este sistema tiene la ventaja de poder ejecutar la aplicación, en la cual estás trabajando en Android Studio, en un dispositivo virtual para probar los diferentes casos de uso y búsqueda de errores.

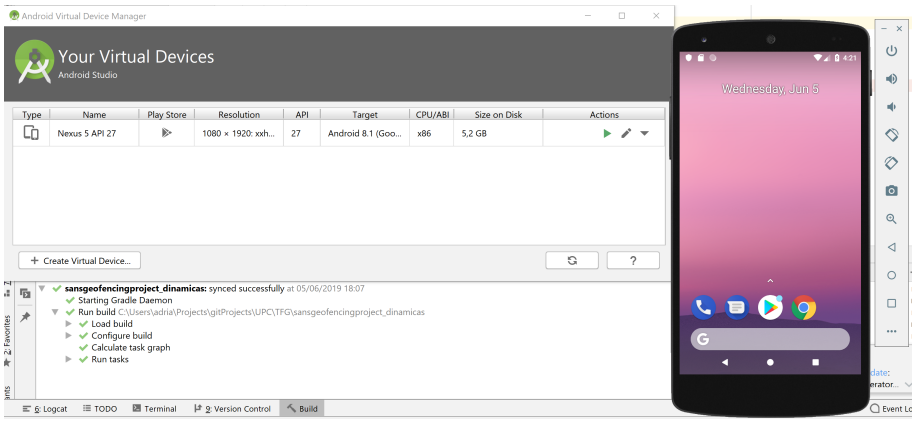


Figura 6.2: Lista de dispositivos virtuales en Android Studio.

Para este proyecto se ha elegido un Nexus 5 con API 2, Android 8.1, para hacer las primeras pruebas en el ordenador

6.4. Desarrollo

6.4.1. Inicio de la estructura del proyecto

Las diferentes características que se van a desarrollar durante esta sección deben ser objeto del seguimiento de un controlador de versiones. Como se comentó en el apartado *Metodología y rigor*, se ha utilizado Git para esta función. La estructura del repositorio debe ser muy estricta para obtener todas los beneficios que te aporta Git.

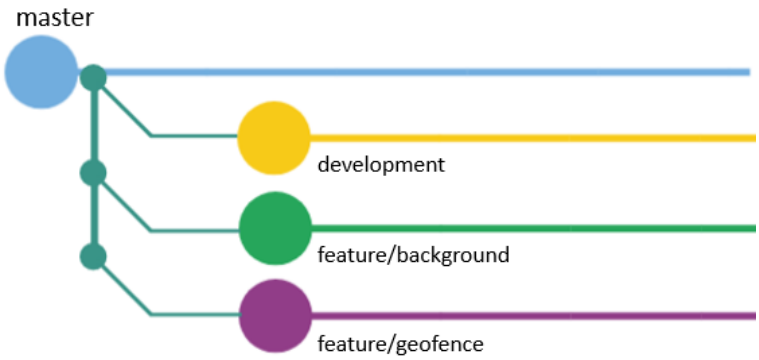


Figura 6.3: Ramas creadas inicialmente. Fuente: elaboración propia.

Para empezar a desarrollar el software de esta sección, se van a crear 3 ramas a partir de la master. Tenemos la rama de development donde se implementará la integración de Geofence y del servicio en segundo plano. Dichas características se van a desarrollar en ramas completamente separadas llamadas feature/background y feature/geofence. En otras palabras, empezaremos desarrollando Geofence y el servicio en segundo plano en la rama correspondiente y cuando estén plenamente en funcionamiento se hará un merge para la integración en la rama de development. Cuando esté todo funcionando, se hará una release final para introducirlo en la rama principal master.

6.4.2. Estudio de la arquitectura a implementar

En esta sección se ha llevado a cabo el estudio de la mejor arquitectura para el propósito que queremos, Geofence ejecutándose como un servicio en segundo plano.

Primeramente se definirán los diferentes diagramas de flujo a seguir, la arquitectura de 3 capas y las APIs a integrar.

Diagramas de flujo

Antes de empezar, se debe pensar qué deben hacer cada una de las implementaciones software que se van a desarrollar. Para ello, se han creado en cada una de las implementaciones sus diagramas de flujo correspondientes. Principalmente se muestra el flujo del programa a seguir.

El objetivo de este apartado es tener una primera idea de la estructura a seguir en el desarrollo software y analizar lo que realmente queremos, y a partir de esto, buscar qué recursos utilizar.

Antes de empezar, se describirá la estructura y comportamientos que comparten cada una de las implementaciones entre sí:

- Petición de los permisos. Para inicializar cualquier aplicación es necesario que el usuario, en este caso yo, acepte los permisos requeridos para el correcto funcionamiento del software. En este caso necesitamos los siguientes permisos:
  - Ubicación.
  - Escritura en almacenamiento interno.
- Comprobación de la creación de los servicios. Después de la creación de los servicios tanto Geofence como segundo plano, es necesario ver si realmente se han creado correctamente. Si fuera el caso de que hubiera algún fallo en la creación del servicio, se volvería a la pantalla principal.
- Capa de persistencia según lo que elija el usuario. El usuario, en este caso yo mismo, antes de hacer el lanzamiento de la aplicación debe ser capaz de seleccionar el método de almacenamiento, Amazon Web Service o el almacenamiento interno del dispositivo.

- Segundo plano

A continuación, se muestra el diagrama de flujo de la implementación de segundo plano:

El motivo de escritura en la capa de persistencia vendrá dado por la ejecución correcta del servicio en segundo plano.

- Geofence en primer plano

A continuación, se muestra el diagrama de flujo de la implementación de Geofence:

Destacar que, el motivo de escritura en la capa de persistencia vendrá dado por la notificación de Geofence.

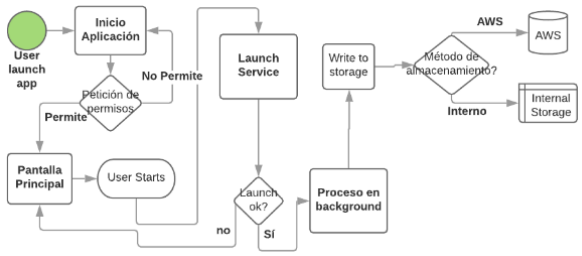


Figura 6.4: Casos de uso de la implementación de segundo plano. Fuente: elaboración propia.

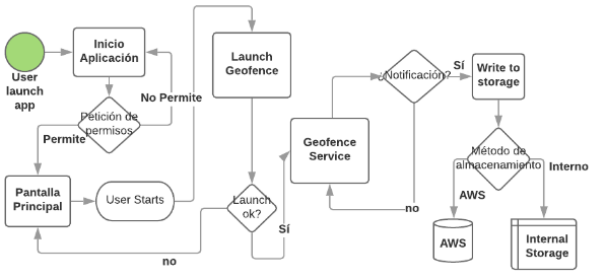


Figura 6.5: Casos de uso de la implementación de Geofence. Fuente: elaboración propia.

- Integración Geofence en segundo plano

A continuación, se muestra el diagrama de flujo de la implementación de la integración de Geofence con el servicio en segundo plano:

Viendo el diagrama de flujo de la integración de los dos servicios, se pueden distinguir cuatro partes diferenciadas del sistema:

- Gestión de los permisos de la aplicación.
- Elección de los parámetros par parte del usuario para empezar la aplicación.
- Ejecución del servicio segundo plano.
- Ejecución de Geofence.

Arquitectura de tres capas

Para el correcto desarrollo de las diferentes implementaciones se ha decidido seguir la metodologia de la arquitectura de tres capas. Consiste en diferenciar

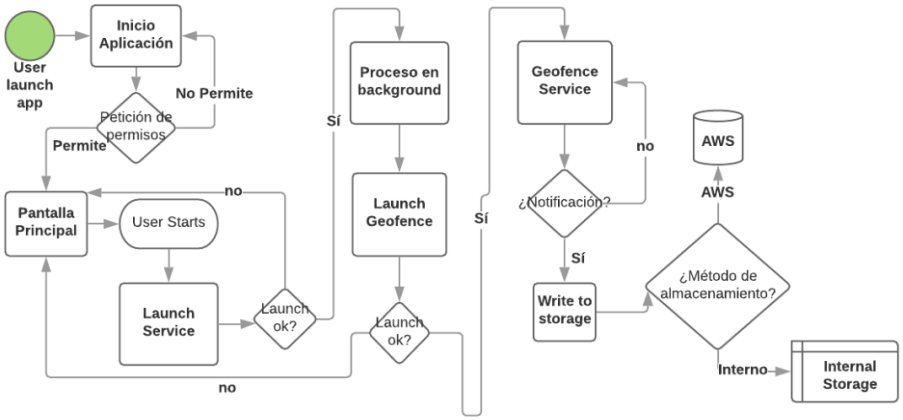


Figura 6.6: Casos de uso de la implementación de la integración. Fuente: elaboración propia.

tres partes de la aplicación que tienen que trabajar independientemente una de la otra con diferentes controladores. Se detalla a continuación:

- Capa de presentación

Esta capa se define cómo la presentación de la aplicación al usuario, en otras palabras, lo que ve el usuario. A través de ella, el usuario va a pasarle las acciones que realice a la capa de dominio.

Estas acciones las vamos a definir como los diferentes parámetros de configuración del sistema, que dependiendo de la implementación, tendrá una u otras.

Parámetro	**Background**	**Geofence**	**Integración**
Radio de las Geofences	-	X	X
Actualizaciones de ubicación	-	X	X
Ejecución en Background o Foreground	-	-	X
Tipo de experimento	-	-	X
Método de almacenamiento	X	X	X

Cuadro 6.1: Dedicación en horas por cada tarea.

El usuario deberá elegir estos parámetros de configuración en la capa de presentación para después enviarlo a la capa de dominio y ejecutar las diferentes funciones a hacer. En el apartado de experimentación se detallará sobre estos parámetros de configuración.

- Capa de dominio

La capa de domino es la que gestiona la lógica de la aplicación, es decir, recibe las solicitudes de la capa de presentación para poder procesarlas y devolver un resultado. Se implementará Geofence y el servicio en segundo plano según los parámetros elegidos por el usuario en la capa de presentación.

En caso de notificación, se ha pensado en crear una clase para poder conectarse con la capa de persistencia, actuando como una controladora para hacer peticiones de escritura, tanto en el almacenamiento interno como en Amazon Web Services.

- Capa de persistencia

Es donde residen los datos de la aplicación, y se encarga de su creación y gestión, tanto de escritura como de lectura. En caso de notificación de la capa de dominio, la controladora escribirá en uno de los parámetros definidos por la capa de presentación como método de almacenamiento.

APIs a integrar en el desarrollo software

En este apartado se pretende mostrar el motivo principal de la elección hecha de cada una de las APIs que se van a integrar en el desarrollo software.

La aplicación presenta 3 características que se pueden aprovechar de diferentes APIs que Android nos ofrece para su utilización, las cuales garantizan un correcto funcionamiento con el sistema operativo Android. Primero, tenemos la característica de localización, segundo la ejecución en segundo plano y finalmente la conexión con Amazon Web Services.

- Localización

En tema de localización, el sistema debe ser capaz de cumplir la presencia de actualizaciones de ubicación y notificaciones provenientes del sistema de Geofence. Para lograr estos dos puntos, Android nos proporciona lo siguiente:

- Fused Location Provider API[3]
- Geofencing API[4]

El motivo de elección de estas dos APIs es bastante sencillo, es lo que Google actualmente recomienda para las aplicaciones basadas en ubicación en aplicaciones ejecutándose en segundo plano. Fused Location se utiliza para obtener localización, ya que, como se ha dicho numerosas veces durante este documento,

[3] https://developers.google.com/location-context/fused-location-provider/
[4] https://developers.google.com/location-context/geofencing/

ofrece una optimización que otras no poseen en referencia a la batería del dispositivo móvil. Por otra banda, tenemos Geofencing API, es la única API que ofrece Google para la integración de la tecnología Geofence en un proyecto, así que es la única opción que se presenta.

- Ejecución segundo plano

La elección de la API a integrar para obtener una ejecución en segundo plano ha sido bastante complicado por las numerosas restricciones que hay actualmente en los sistemas operativos Android.

Para ver el proceso de elección de JobScheduler API[5], ver las diferentes explicaciones de las figuras 5.7 y **??**.

JobScheduler necesita de la clase jobInfo[6], que se define como un contenedor de datos de configuración del servicio en segundo plano, el cual se debe introducir en la creación de jobScheduler.

- Amazon Web Services

La integración de Amazon Web Service en un proyecto basado en Android es algo que ya estaba desarrollado en el grupo de investigación el cual estoy trabajando actualmente, y es por eso motivo que no se ha entrado en detalle de su desarrollo.

Para garantizar la comunicación entre Android y AWS se utilizan ApiClientFactory y CognitoUserPool. CognitoUserPool se utiliza para que el usuario de la aplicación se pueda autentificar con AWS, en este caso se ha puesto en el proyecto Android un nombre de usuario y contraseña ya configurados en AWS. Finalmente se usa ApiClientFactory para la comunicación de ambas partes.

6.4.3. ¿Qué datos te proporciona Geofence?

Cuando se produce una notificación de transición gracias al servicio Geofence, la API te proporciona diferentes campos de información que puedes obtener para ver lo que realmente a sucidido.

- Tipo de transición.
- Una lista de las Geofences donde se ha producido esa transición.
- Objeto Location. Para más detalle ver sección *Formato de almacenamiento*

[5] https://developer.android.com/reference/android/app/job/JobScheduler
[6] https://developer.android.com/reference/android/app/job/JobInfo

6.4.4. Desarrollo del servicio en segundo plano

El desarrollo del servicio en segundo plano es fundamental que funcione correctamente. Es uno de los requisitos de este proyecto y su funcionamiento depende de los otros, ya que, sin un servicio en segundo plano, no podemos obtener los datos para proceder a su análisis.

Después de la elección del sistema a integrar, jobScheduler API, se ha procedido a crear el diagrama UML correspondiente a este apartado. La creación de un diagrama UML permite ver la comunicación necesaria de cada clase y tener una representación gráfica de la solución al problema.

Diagrama UML

En el siguiente diagrama se puede observar los diferentes parámetros y métodos implementados en cada una de las clases que se han desarrollado.

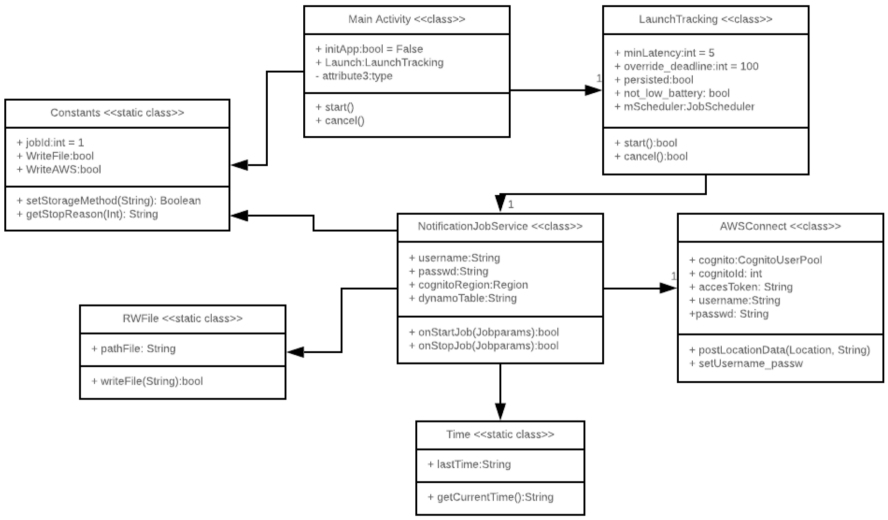


Figura 6.7: Diagrama UML de la implementación de segundo plano.

A continuación, en el siguiente apartado se detallarán cada una de las clases que se muestran en el diagrama UML.

Descripción de las clases implementadas

- Main Activity

Esta clase es la responsable de avisar a la clase Launch Tracking de la puesta en marcha del servicio en segundo plano. Para ello, antes se le permite al usuario

elegir el método de almacenamiento de los resultados del servicio en segundo plano, AWS o almacenamiento interno del dispositivo móvil.

Cuando se crea el servicio, esta clase también se encarga de enviar el parámetro de configuración a la clase Constants y de comprobar que el servicio ha sido creado correctamente. En caso contrario, la aplicación informa al usuario mediante un Toast[7] de que el servicio en segundo plano no ha sido creado.

- Launch Tracking

Esta clase es invocada por la MainActivity, y se encarga de la creación del servicio en segundo plano y de la devolución a la MainActivity de su correcta creación.

A continuación se muestra un extracto de código de la creación del servicio en segundo plano, el cual empezará ejecutando la clase NotificationJobService en la función onCreate().

```
1  /*
2  * Schedule a jobService starting with NotificationJobService class
3  */
4
5  ComponentName componentName = new ComponentName(context,
       NotificationJobService.class);
6
7  JobInfo.Builder builder = new JobInfo.Builder(JOB_ID, componentName);
8  builder.setMinimumLatency(MIN_LATENCY * 1000)
9         .setOverrideDeadline(OVERRIDE_DEADLINE * 1000)
10        .setPersisted(PERSISTED);
11
12 if (Build.VERSION.SDK_INT >=26) {
13     builder.setRequiresBatteryNotLow(NOT_LOW_BATTERY);
14 }
15
16 mScheduler = (JobScheduler) context.getSystemService(JobScheduler.class);
17 int success = mScheduler.schedule( builder.build());
18
19 if (success == JobScheduler.RESULT_SUCCESS) return true;
20 else return false;
```

Destacar el control de la creación del servicio devolviendo falso en caso que el resultado de creación del jobScheduler no sea exitoso.

- NotificationJobService

El objetivo de esta clase es solamente escribir en un fichero, ya que queremos ver que realmente el jobService se está ejecutando en segundo plano realmente.

Esta clase se extiende de la clase JobService, la cual contiene las funciones onStartJob y onStopJob. Estas dos funciones son ejecutadas cuando el servicio es iniciado o terminado, y cada vez que se ejecute escribirá en el fichero interno

[7] https://developer.android.com/guide/topics/ui/notifiers/toasts

del dispositivo móvil la hora en que se ejecutan. Con esto sabremos cuánto dura de media un jobService.

- AWSConnect

AWSConnect está capacitada para escribir en AWS, más concretamente en una tabla definida dentro de la dynamoDB[8].

- Constants

Esta clase se define como una clase estática, en otras palabras, podemos acceder a esta clase sin necesidad de que el objeto de la clase sea instanciado. Esto nos facilita consultar los parámetros de configuración del usuario en la pantalla principal.

- RWFile

Para garantizar la persistencia, esta clase es capaz de escribir en un fichero interno del teléfono móvil para guardar todos los datos necesarios para los diferentes análisis que se detallan en el apartado *Análisis de datos*.

A continuación se presenta el extracto de código el cual hace posible su funcionalidad.

```
1  /**
2       * Writes geofence information to a file located in documents folder
3       * @param data Text message to write
4   */
5
6  public static void writeFile(String data) throws IOException {
7
8      Date c = Calendar.getInstance().getTime();
9      SimpleDateFormat df = new SimpleDateFormat("MM-dd-yyyy", Locale.US);
10     String date = df.format(c);
11
12     // Get the directory for the user's public document directory.
13     final File path = Environment.getExternalStoragePublicDirectory (
           Environment.DIRECTORY_DOCUMENTS + "/GeofenceData/");
14     if(!path.exists()) {
15         path.mkdirs();
16     }
17
18     final File file = new File(path, "data" + date + ".txt");
19
20     try
21     {
22         file.createNewFile();
23         FileOutputStream fOut = new FileOutputStream(file,true);
24         OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
25         myOutWriter.write(data);
26         myOutWriter.close();
27
28         fOut.flush();
29         fOut.close();
```

[8]https://aws.amazon.com/es/dynamodb/

```
30      }
31      catch (IOException e)
32      {
33          Log.e("Exception", "File write failed: " + e.toString());
34      }
```

El fichero de escritura se guardará en una carpeta llamada "GeofenceData"dentro de la carpeta de documentos del dispositivo móvil.

- Time

Esta clase se define como clase estática, y solamente contiene la funcionalidad de obtener la hora actual. ¿Por qué se necesita esta función? Por dos motivos:

- Controlar la hora de inicio y finalización del servicio en segundo plano. Con esto podemos comparar cuato tiempo dura de media la vida de un servicio en segundo plano.
- Puede darse el caso de que una notificación de Geofence te proporcione Location como valor nulo, en otras palabras, no obtienes latitud, longitud ni la hora de la notificación de Geofence. Es por eso que debemos controlar el caso en el que sea necesaria la fecha de notificación si Location es nulo.

```
1
2   public static String getCurrentTime()
3   {
4       Date c = Calendar.getInstance().getTime();
5       SimpleDateFormat df = new SimpleDateFormat("MM-dd-yyyy'T'HH:mm:ss",
            Locale.US);
6       return df.format(c);
7   }
```

El formato elegido para obtener la hora exacta es "MM-dd-yyyy'T'HH:mm:ss". Por ejemplo hoy día 03/06/2019 con hora 10:27:12 se escribiría como: 06-03-2019T10:27:12.

6.4.5. Desarrollo de Geofence

El desarrollo de Geofence es fundamental, ya que su correcto funcionamiento es el principal requerimiento definido en el proyecto, con el objetivo de la recogida de datos y su posterior análisis.

Después de explicar que se va a utilizar GeofenceAPI, se procede a la creación del diagrama UML para permitir ver la comunicación necesaria de cada clase y tener una representación gráfica de la solución al problema.

Diagrama UML

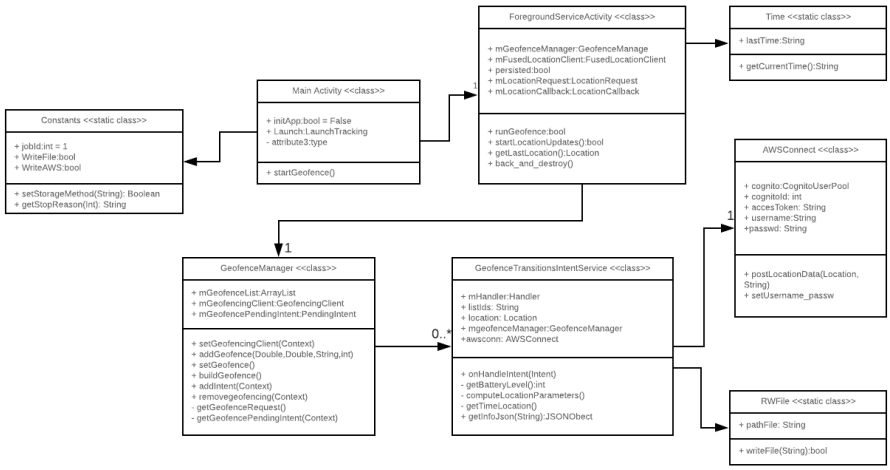


Figura 6.8: Diagrama UML de la implementación de Geofence.

Descripción de las clases implementadas

- Main Activity

En este caso, los parámetros preguntados al usuario serán sobre la configuración de Geofence y del proveedor de ubicación Fused Location. Tal como se detalla en la tabla 7.13, se preguntará sobre el radio de las Geofences, si queremos o no actualizaciones de ubicación y, igual que el desarrollo en segundo plano, el método de almacenamiento.

Una vez escogidos cada unos de los parámetros, se procederá a la creación de la Actividad ForegroundServiceActivity, la cual definirá el servicio de Geofencing.

- ForegroundServiceActivity

Esta clase es la encargada de configurar el proveedor de servicio de Fused Location en relación a los parámetros escogidos por el usuario en la pantalla principal. Además se procede a la construcción del servicio de Geofencing a partir de la clase GeofenceManager. A continuación, se enseñarán fragmentos de código de las dos configuraciones.

- Función run_geofence.

Seguidamente, se muestra el código de la construcción del servicio de Geofence a partir de la clase GeofenceManager.

Destacamos el control de errores en la creación de Geofence cuando añadimos los puntos de interés y la obtención de una muestra de localización justo después de la creación de Geofence, siempre y cuando, el usuario haya elegido la presencia de actualizaciones de ubicación.

```
geofenceManager= GeofenceManager.getInstance();
geofenceManager.setContext(this.getApplicationContext());
geofenceManager.setGeofencingClient(this.getApplicationContext());
Boolean ok = geofenceManager.setGeofencePOIS();
if (!ok) return;

// create geofence
geofenceManager.buildGeofence();
geofenceManager.addIntent(this.getApplicationContext());

if (Constants.LOCATION_UPDATES_ON) getLastLocation();
```

- Función startLocationUpdates.

Es llamada en la funcion onResume de la actividad que define el servicio en foreground, y es capaz de empezar a hacer las actualizaciones de ubicación. Cabe destacar que la actualización de ubicación no las recogemos, ya que la única utilidad que hace esto es que trabaje conjuntamente con Geofece, con esto quiero decir que, Geofence internamente cogerá las ubicaciones solicitadas por esta actividad y las aprovechará para entregar las notificaciones que crea convenientes.

Los diferentes parámetros de configuración de Locationrequest, se detallarán en el siguiente capítulo.

```
protected void onResume() {
    super.onResume();
    if (Constants.LOCATION_UPDATES_ON) {
        startLocationUpdates();
    }
}
```

```
private void startLocationUpdates() {
    if (mFusedLocationClient == null) {
    // get the Fused location provider client
    mFusedLocationClient = LocationServices.getFusedLocationProviderClient(
        this);
    }
    if (mLocationRequest == null) {
        mLocationRequest = LocationRequest.create()
                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
                .setInterval(Constants.FASTEST_INTERVAL_FOREGROUND * 1000L)
                .setFastestInterval(Constants.FASTEST_INTERVAL_FOREGROUND *
                    1000L)
                .setMaxWaitTime (Constants.INTERVALS * 1000L * 100);
    }
```

```
13
14      mLocationCallback = new LocationCallback() {
15          @Override
16          public void onLocationResult(LocationResult locationResult) {
17              //Nothing to do
18          }
19      };
20
21      mFusedLocationClient.requestLocationUpdates(mLocationRequest,
            mLocationCallback,null);
22  }
```

- GeofenceManager

Esta clase se encarga de la gestión del servició Geofence: crea cada uno de los círculos, añade el Intent para las notificaciones, construye el servicio y controla los errores.

Las funciones más importantes en esta clase son las siguientes:

1. buildGeofence(). Es la creación del servicio de Geofence a partir de sus parámetros de configuración, los cuales serán explicados en el siguiente capítulo. Como se ve en el código, se define Geofence con las transiciones de entrada, salida y espera.

```
1   /**
2    * Build geofence with {@link Geofence.Builder}.
3    */
4   public void buildGeofence(){
5       int i;
6       int MAX_GEO = geofence_lats.size();
7
8       for (i = 0; i < MAX_GEO; i++) {
9           mGeofenceList.add(new Geofence.Builder()
10              .setRequestId(geofence_names.get(i))
11              .setCircularRegion(
12                      geofence_lats.get(i),
13                      geogence_longs.get(i),
14                      geofence_radius.get(i)
15              )
16              .setExpirationDuration(Geofence.NEVER_EXPIRE)
17              .setLoiteringDelay(Constants.LOITERING_DELAY * 1000)
18              .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
19                      Geofence.GEOFENCE_TRANSITION_EXIT |
20                      Geofence.GEOFENCE_TRANSITION_DWELL)
21              .setNotificationResponsiveness(Constants.
                    NOTIFICATION_RESPONSIVENESS * 1000)
22              .build());
23      }
24  }
```

2. addIntent(). Es el momento de la verdadera puesta en marcha de Geofence dentro del dispositivo móvil y es por eso que necesita diferentes controles de errores. Primeramente, consultamos que los permisos en relación a la ubicación han sido aceptados por el usuario. También, se puede ver que después de

añadir el Intent de Geofence, se avisa sobre el estado de la puesta en marcha de Geofence, si ha sido correcto o no. Según el estado de después de la creación, se mostrará al usuario como 'geofence added' o 'failed to create geofence'.

```
/**
 * Adds Intent from geofence pending Intent to geofencing client.
 * @param context context
 */
public void addIntent(final Context context){
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M ||
        (ContextCompat.checkSelfPermission(context, Manifest.permission.
            ACCESS_COARSE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) &&  // AND!!
        (ContextCompat.checkSelfPermission(context, Manifest.permission.
            ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED)) {

        return;
    } else {
        mGeofencingClient.addGeofences(getGeofencingRequest(),
            getGeofencePendingIntent(context))
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    // Geofences added
                    if (Constants.isDEBUGGING()) {
                        Log.d(TAG, "Geofence added");
                        Toast.makeText(context.getApplicationContext(), "
                            Geofence added", Toast.LENGTH_LONG).show();
                    }
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    String errors = "-1";
                    if (e instanceof ApiException) {
                        errors = Constants.getErrorStringGeofence(((
                            ApiException) e).getStatusCode());
                    }
                    if (Constants.isDEBUGGING()) {
                        Log.e(TAG, "Failed to create geofence");
                        Toast.makeText(context.getApplicationContext(), "
                            Error adding Geofence: " + errors, Toast.
                            LENGTH_LONG).show();
                    }
                }
            });
    }
}
```

- GeofenceTransitionIntentService

Esta clase es la que obtiene el Intent de notificación de Geofence, la cual extiende de IntentService utilizando el método onHandleIntent(Intent intent), donde Intent es recogido para obtener el evento de Geofence. En el fragmento de código que se muestra a continuación se muestra como es el tratamiento de este evento:

```
protected void onHandleIntent(Intent intent) {

    GeofencingEvent geofencingEvent = GeofencingEvent.fromIntent(intent);

    if (Constants.getWRITE_AWS()) {
        awsconn = AWSConnect.getInstance();
        awsconn.setDynamoDBTableGeofence(dynamoTableGeofence);
    }
    if (Constants.getWRITE_FILE()) {
        try {
            RWFile.writeToDocument_GeofenceData("Notification time OS: " +
                Time.getCurrentTime() + "\n");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    if (geofencingEvent.hasError()) {
        int errorCode = geofencingEvent.getErrorCode();
        String errorEvent = Constants.getErrorStringGeofence(errorCode);
        Log.w(TAG, "Geofence event has Error.");
        if (Constants.getWRITE_FILE()) {
            try {
                RWFile.writeToDocument_GeofenceData("Geofence Event has error
                    : " + errorEvent + "\n" );
            } catch (IOException e) {
                Log.e("Exception", "File write failed: " + e.toString());
            }
        }
        return;
    }

// Get the transition type.
final int geofenceTransition = geofencingEvent.getGeofenceTransition();

if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER ||
        geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT ||
        geofenceTransition == Geofence.GEOFENCE_TRANSITION_DWELL) {

    // Get the geofences that were triggered.
    List<Geofence> triggeringGeofences = geofencingEvent.
        getTriggeringGeofences();

    if ( geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER) {
        listIds= "ENTER#";
    }
    if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {
        listIds = "EXIT#";
    }
    if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_DWELL) {
        listIds = "DWELL#";
    }
    //A single event can trigger multiple geofences.
    for (Geofence geofence : triggeringGeofences) {
        String locationId = geofence.getRequestId();
        listIds += "*" + locationId;
    }

    Log.i(TAG, "Geofence transition done: " + listIds);

    location = geofencingEvent.getTriggeringLocation();
```

Toda este fragmento de código muestra la manera de ver como se recoge la

información de la notificación de Geofence a partir del Intent obtenido. Cabe decir que acaba con la adquisición del objeto Location, el cual nos hace falta para obtener la longitud y latitud del punto donde se produce el evento de transición.

Primeramente, se utiliza el método de consulta hasError(), para poder acabar la función lo antes posible en caso de producirse un error en un momento determinado del evento. Después de esto, ya es posible recoger toda la información, empezando por el tipo de transición EXIT, ENTER o DWELL; además, el identificador de las Geofences que han provocado dicho evento. Después de esto, viene el getter del objeto Location para que más adelante se extraiga más información de él, como por ejemplo, lo comentado anteriormente, la longitud y latitud del evento en sí. Este apartado de extracción de información del objeto Location se explicará en el siguiente capítulo.

- AWSConnect

Su configuración e implementación es prácticamente igual al desarrollo del servicio en segundo plano, y es por eso que no veo conveniente detallarlo otra vez.

- Constants

En esta clase se añaden los diferentes parámetros de configuración que el usuario ha escogido en la pantalla principal, y al ser una clase estática no es necesario instanciar la clase en caso de consultar cualquier parámetro.

- RWFile

Su configuración e implementación es prácticamente igual al desarrollo del servicio en segundo plano, y es por eso que no veo conveniente detallarlo otra vez.

- Time

A parte de ya lo explicado en la sección anterior, también apuntaremos la hora de inicio del servicio de Geofencing, que debería ser parecida a la del inicio del servicio en primer plano.

6.4.6. Integración de Geofence como servicio en segundo plano

Después del correcto funcionamiento de las dos implementaciones anteriores, ahora se explicará la integración de Geofence como un servicio de segundo plano. Primeramente decir que los cambios en el código han sido escasos, solamente

se han tenido que mover algunas relaciones, mostradas en el siguiente diagrama UML

Diagrama UML

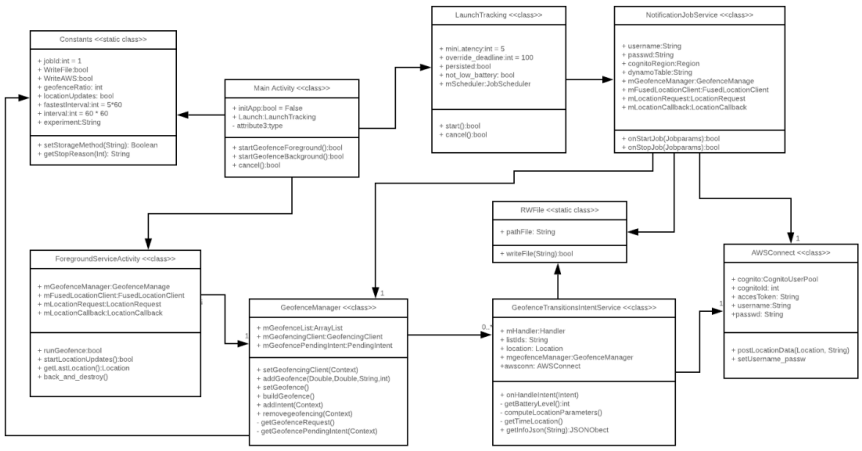


Figura 6.9: Diagrama UML de la integración.

Lo más destacable es el cambio de Geofence en el servicio de segundo plano. Esto lo hemos hecho cambiando todas las llamadas a GeofenceManager de la clase ForegroundServiceActivity a la función onCreate de la clase NotificationJobService, para que Geofence sea lo primero que se cree cuando se ejecute por primera vez el servicio en segundo plano.

A parte de esto, y como se puede ver en el diagrama UML, durante el desarrollo del proyecto, se ha decidido dejar en la implementación final la clase ForegroundServiceActivity ya que queremos ver cuál es el funcionamiento de Geofence en una actividad en primer plano.

Diferencias en la integración respecto a las otras dos implementaciones

Este aparatado no es muy extenso, ya que en la integración solamente ha habido cambio de código de una clase a otra. A continuación explicaremos que más importante en los cambios.

- Clase NotificationJobService crea Geofence. Todo el trozo de código visto en ForegroundServiceActivity ha pasado directamente NotificationJobSer-

vice, ya que es el servicio de segundo plano quien debe crear el Geofence, siempre y cuando el usuario lo elija.

- Todos los parámetros de configuración se encuentran en la pantalla, y cuando el usuario inicia el algoritmo, los valores de los parámetros se guardan en la clase Constants como estáticos, para que cualquier clase que lo necesite acceda a ellos. Como he dicho, se destaca la elección de Geofence ejecutándose en segundo o primer plano. Una vez elegidos todos los parámetros, el usuario debe darle al botón Start y esperar a que se muestre el Toast de que Geofence se ha añadido correctamente.

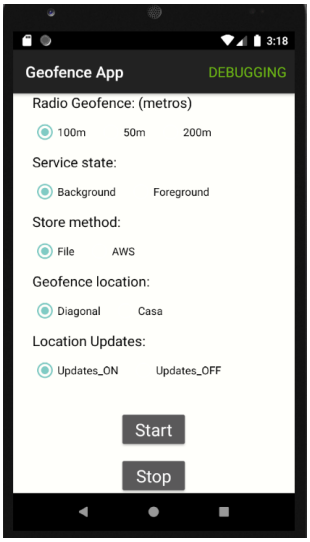


Figura 6.10: Pantalla principal del desarrollo de integración. Parámetros de configuración.

6.5. Análisis

En esta sección se ha cumplido uno de los principales objetivos de este proyecto, el desarrollo de un aplicación que sea capaz de tener un servicio en segundo plano recibiendo notificaciones de Geofences.

Realmente tenemos que sacar conclusiones de este apartado y ver a partir del análisis la sección *análisis de datos*, lo que realmente se ha detectado sobre el comportamiento de Geofence y el servicio en segundo plano en el sistema operativo Android.

En el siguiente capítulo, antes de definir los diferentes experimentos, se verá cómo hemos estructurado la persistencia de los datos, para luego obtener un

análisis de ellos, y qué hemos visto para la corrección de errores y de configuraciones de este apartado. Con esto quiero decir que este apartado se ha desarrollado en paralelo al siguiente, ya que necesitamos realmente analizar muy bien los datos obtenidos de la última implementación, para ver que realmente no existen errores de código ni de configuración.

Dicho esto, cabe decir que toda esta sección viene dada por el resultado final de la arquitectura y de la implementación, las cuales se han obtenido después de los análisis de datos y corrección de errores del siguiente apartado.

Capítulo 7

Captura de datos, experimentación y análisis.

7.1. Introducción

Actualmente se encuentra muy poca información del comportamiento real de Geofence, tanto en servicios de segundo plano como el en primer plano, siendo así, el motivo de la realización de este proyecto. En este capítulo se detalla cómo funciona este servicio.

Como hemos visto en el capítulo anterior, se ha implementado el sistema Geofence dentro de un servicio en segundo plano, y para poder garantizar su correcto funcionamiento, primeramente se debe definir qué datos se deben obtener de la aplicación.

Obtenidos estos datos, se efectuará un análisis con el objetivo de corregir de errores de la implementación ya construida. Después se definirán los diferentes experimentos que servirán para ver el comportamiento de Geofence.

A continuación, cuando tengamos los experimentos bien definidos, se mostrarán tanto los diferentes resultados de cada uno de ellos como las diferentes conclusiones que podemos obtener. Y para acabar con este capítulo, se hará un análisis general de los resultados para ver realmente cómo se comporta Geofence delante de diferentes situaciones.

Una vez obtenidas las diferentes conclusiones del comportamiento de Geofence, tendremos una información que poca gente sabe. Es por eso que este trabajo será objeto de la extracción de información para publicar artículos de investigación sobre Geofences y los servicios en segundo plano en los dispositivos Android actuales.

Cabe destacar, y es algo que se puede ver en el apartado *Metodología y rigor*, las dos tareas que recogen este capítulo se han desarrollado en paralelo con el anterior, ya que la implementación ha ido progresando positivamente con el

análisis de los datos obtenidos y corrección de errores.

7.2. Objetivos

Los objetivos de esta parte se pueden resumir en:

- Definición de la estructura de datos obtenidos de la aplicación.
  - Deben ser claros y concisos.
- Corrección de errores de la implementación y estructura software a partir de diferentes análisis de datos.
- Definición de los diferentes experimentos a ejecutar.
  - Deben proporcionar datos para efectuar un análisis.
  - Los experimentos deben ser suficientemente importantes para cumplir con el objetivo de entender el verdadero comportamiento de Geofence.
- Analizar los datos obtenidos de los resultados de cada unos de los experimentos efectuados.

7.3. Recursos

7.3.1. Python

Figura 7.1: Logotipo de Python.

Python es un lenguaje de programación muy sencillo de usar, potente y de código abierto, el cual pone mucho énfasis en tener una sintaxis altamente leíble. Para su ejecución, se ha utilizado la plataforma Spyder[1], es un entorno de programación que se utiliza en el grupo de investigación el cual estoy trabajando, ya que proporciona un entorno para la programación en el análisis de datos.

[1] https://www.spyder-ide.org/

Se ha elegido este lenguaje de programación por el simple hecho de su facilidad en el tratamiento de datos. Se utilizará para generar un fichero en JavaScript que muestre los diferentes datos en un mapa y para el tratamiento de datos de Geofence.

7.3.2. JavaScript

Es un lenguaje de programación que te permite efectuar operaciones complejas implementadas como una parte de un navegador web, es decir, trabaja conjuntamente con html creando contenido nuevo y dinámico.

Figura 7.2: Logotipo de JavaScript.

En este proyecto, el código JavaScript será generado a partir de un programa Python, creará un fichero con la extensión .js, especifico de javascript, capaz de interactuar con un mapa definido en un HTML, obtenido de la biblioteca Leaflet.

Leaflet

Leaflet es una biblioteca de JavaScript de código abierto que te permite interactuar con mapas. Contiene una API fácil de usar y muy bien documentada, con todas las funciones que la mayoría de desarrolladores necesitan.

Figura 7.3: Logotipo de Leaflet.

Su utilización en este proyecto es proporcionarnos un mapa donde podamos marcar cada una de las notificaciones proporcionadas por el servicio de Geofence y, además, para entender el significado de cada notificación, se dibujará cada una de las Geofences que hayan sido definidas en esa ejecución.

Gracias a esta API, podemos ver visualmente los datos que obtenemos de la aplicación y cuál es su significado dentro de un mapa.

7.3.3. HTML

Es un lenguaje para dar una estructura al contenido web, tal como, se definen las cabeceras, imágenes, tablas, párrafos,etc. En nuestro caso, nuestro HTML definirá un página web proporcionada por la biblioteca de Leaflet, donde también se integrarán los diferentes JavaScripts de las notificaciones y Geofences definidas.

A continuación, se puede ver la integración de Leaflet en un formato html. Además, podemos ver como se han incluido los ficheros JavaScript de las Geofences definidas y las notificaciones, creados ambos, por el programa Python.

```
1
2  <!DOCTYPE html>
3  <html>
4  <head>
5    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.1/dist/leaflet.
         css" integrity="sha512-Rksm5RenBEKSKFjgI3a41vrjkw4EVPlJ3+
         OiI65vTjIdo9brlAacEuKOiQ5OFh7cOI1bkDwLqdLw3Zg0cRJAAQ==" crossorigin=""
         />
6    <script src="https://unpkg.com/leaflet@1.3.1/dist/leaflet.js" integrity="
         sha512-/Nsx9X4HebavoBvEBuyp3I7od5tA0UzAxs+
         j83KgC8PU0kgB4XiK4Lfe4y4cgBtaRJQEIFCW+oC506aPT2L1zw==" crossorigin=""
         ></script>
7
8  </head>
9  <body>
10   <div id="mapDiv" style="width: 1400px; height: 800px"></div>
11   <script>
12     var lat =41.38794035450985;
13     var lon = 2.11227;
14     // initialize map
15     map = L.map('mapDiv').setView([lat, lon], 13);
16     // set map tiles source
17     L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
18       attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">
             OpenStreetMap</a> contributors',
19       maxZoom: 18,
20     }).addTo(map);
21   </script>
22   <script src="output_geofences.js"></script>
23   <script src="output_notifications.js"></script>
24 </body>
25
26 </html>
```

7.4. Desarrollo

7.4.1. Estructura de almacenamiento de los datos

Antes de cualquier análisis de datos, es preferible que se definan los diferentes campos de obtención de la información, proporcionados por los diferentes servicios que se ejecutan en la aplicación móvil.

En este apartado se presentará cuál es el formato de obtención de los datos de ubicación generados por nuestra implementación software instalada en un

dispositivo móvil. Por otro lado, el formato del almacenamiento también influirá en esa decisión, ya que por ejemplo, almacenando los datos internamente en el teléfono nos permite una mayor flexibilidad para ver en tiempo real que realmente se generan datos provenientes de la aplicación.

Lugar de almacenamiento interno.

El espacio del teléfono móvil donde se guardaran los datos obtenidos de la aplicación será dentro del directorio de documentos del dispositivo móvil. Dentro de esta carpeta se creará, si no se ha creado anteriormente, una carpeta llamada GeofenceData donde se guardaran los diferentes ficheros que contienen los datos.

```
1  Date c = Calendar.getInstance().getTime();
2  SimpleDateFormat df = new SimpleDateFormat("MM-dd-yyyy", Locale.US);
3  String date = df.format(c);
4
5  // Get the directory for the user's public pictures directory.
6  final File path = Environment.getExternalStoragePublicDirectory (Environment.
       DIRECTORY_DOCUMENTS + "/GeofenceData/");
7  if(!path.exists()) {
8      path.mkdirs();
9  }
10 final File file = new File(path, "data" + date + ".txt");
```

Además, para garantizar el seguimiento de cada uno de los ficheros creados, su nombre será definido por el día de ejecución de la aplicación. Por ejemplo, si hoy es día 03/06/2019, el fichero de datos será llamado "data06-03-2019.txt", y todas las ejecuciones de este día se guardaran en el mismo.

Formato de almacenamiento

Cuando una notificación Geofence llega al servicio, sea en segundo o primer plano, nuestro sistema debe ser capaz de escribir todos los datos que te proporciona Geofence en un fichero interno o en Amawon Web Services.

La adquisición de los datos de la notificación, en todos los casos, se empaquetarán en formato JSON para facilitar el intercambio de datos. Y antes de este empaquetamiento, se deben tener en cuanta dos casos: Recibir el objeto Location siendo nulo o no nulo.

A continuación se muestra el extracto de código que obtiene los datos de la notificación de Geofences cuando Location es no nulo:

```
/**
     * Creates Json object with Location event
     * @param listIds Contains Geofence events
     * @param locationParameters Contains the information about Location
     * @return JSONObject Contains the information about Geofence triggered
     */
    private JSONObject getJsonLocationGeofence(String listIds, String
         locationParameters) throws JSONException {

        String sDate = getTimeLocation();

        JSONObject jsonItem = new JSONObject();
        jsonItem.put("Id", "-");
        jsonItem.put("DateTime", sDate);
        jsonItem.put("event", listIds);
        jsonItem.put("information", locationParameters);
        jsonItem.put("latitude", Double.toString(location.getLatitude()));
        jsonItem.put("longitude", Double.toString(location.getLongitude()));
        jsonItem.put("token", "-");
        jsonItem.put("BatteryLevel", Integer.toString(getBatteryLevel()));

        return jsonItem;
    }
```

Primeramente obtenemos la hora exacta de la notificación, la longitud y latitud

- Hora exacta del sistema operativo cuando se produce la notificación.
- Hora en la que se produce la transición.
- Latitud y longitud del evento.
- Listids: es un String que contiene el evento de transición de Geofence. Consta del tipo de la transición (entrada, salida o espera) y de los diferentes Geofences donde se ha producido dicha transición.
- BatteryLevel: nivel de batería en el momento de la notificación.
- locationParameters: Una vez obtenemos el objeto Location, se puede obtener la siguiente información:

```
    /**
     * Get parameters from Location
     */
    @RequiresApi(api = Build.VERSION_CODES.O)
    private String computeLocationParameters() {
        String accuracy = (location.hasAccuracy()) ? String.valueOf(
            location.getAccuracy()) : "-";
        String altitude = (location.hasAltitude()) ? String.valueOf(
            location.getAltitude()) : "-";
        String bearing = (location.hasBearing()) ? String.valueOf(
            location.getBearing()) : "-";
        String provider = String.valueOf(location.getProvider());
        String speed = (location.hasSpeed()) ? String.valueOf(location.
            getSpeed()) : "-";

        String bearingAccuracy = (location.hasBearingAccuracy()) ?
            String.valueOf(location.getBearingAccuracyDegrees()) : "-";
        String speedAccuracy = (location.hasSpeedAccuracy()) ? String.
            valueOf(location.getSpeedAccuracyMetersPerSecond()) : "-";
```

```
14          String verticalAccuracy = (location.hasVerticalAccuracy()) ?
                String.valueOf(location.getVerticalAccuracyMeters()) : "-";
15
16          String locationParameters = provider + "#" + accuracy + "#" +
                altitude + "#" + bearing + "#" + bearingAccuracy + "#" +
                speed + "#" + speedAccuracy + "#" + verticalAccuracy;
17          return locationParameters;
18      }
```

Lo más interesante que podemos ver aquí es que podemos obtener el proveedor de la localización y la exactitud total de los datos.

Cuando se dé el caso de obtener el objeto Location como un valor nullo, solamente podemos obtener el tipo de transición que se ha efectuado y las Geofences que lo han provocado. Se muestra en el siguiente fragmento de código:

```
1  /**
2   * Creates Json object only with the event, location is null
3   * @param listIds Contains Geofence events
4   * @return JSONObject Contains the information about Geofence triggered
5   */
6  private JSONObject getJsonWithoutLocationGeofence(String listIds) throws
       JSONException {
7
8      JSONObject jsonItem = new JSONObject();
9      jsonItem.put("Id", "-");
10     jsonItem.put("DateTime", Time.getCurrentTime());
11     jsonItem.put("event", listIds);
12     jsonItem.put("token", "-");
13     jsonItem.put("BatteryLevel", Integer.toString(getBatteryLevel()));
14
15     return jsonItem;
16 }
```

Con todo esto explicado, ya se tiene constancia de cómo y con qué formato se guardan los diferentes datos de ubicación obtenidos de la aplicación software. Además, cada vez que se inicie GeofenceManager o el servicio de segundo plano se cancele, se escribirá en el fichero interno del teléfono la hora del evento producido.

7.4.2. Análisis de datos de las primeras puestas en marcha

Para la realización de esta sección, se ha elegido ejecutar la aplicación durante el trayecto de mi casa hasta la universidad. Es un camino que hago la mayoría de días, y me es muy practico para probar la aplicación.

De esta manera, gracias a la ejecución de este trayecto, se pueden analizar los datos obtenidos e ir corrigiendo los diferentes errores de código o de arquitectura en el momento que llegas al destino.

Recorrido y Geofences definidas.

A continuación se muestra el mapa definido para los análisis durante el desarrollo de las implementaciones basadas en Geofence. Se puedem apreciar dos caminos diferentes, uno que hago yo andando y otro que hago en metro, y alrededor de ellos se encuentran todas las Geofences definidas de forma estática en el código. Los resultados de localización de Geofence con esta configuración deberán ser parecidos a los caminos mostrados.

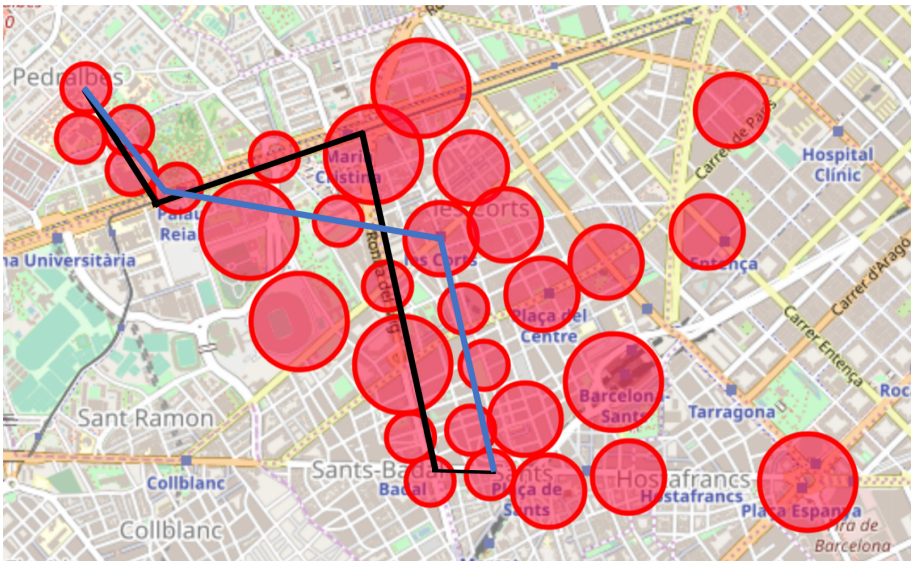


Figura 7.4: Mapa para los primeros análisis de Geofences.

Viendo el mapa, se puede ver que es bastante idóneo, ya que las Geofences definidas se encuentran alrededor de los dos recorridos habituales para llegar a la Universidad, y es por eso que, este mapa se utilizará para ver si las diferentes implementaciones iniciales producen transiciones en las diferentes áreas.

Ejecuciones y primeras impresiones sobre Geofence

Implementación segundo plano

Durante el desarrollo de esta implementación fue un poco larga por la falta de experiencia del desarrollador, pero una vez entender todo el entorno de trabajo de Android Studio, se cumplió con el objetivo.

Durante la ejecución del programa, se crea un servicio en segundo plano

ejecutándose en paralelo a la aplicación en primer plano, capaz de escribir en un fichero.

Durante algunas ejecuciones, vimos que el servicio se va reiniciando cada 10 minutos, en otras palabras, cada 10 minutos obteníamos una escritura en el fichero interno. Después de unos días de investigación, vimos que el servicio se cancelaba después de estar ejecutándose 10 minutos con un timeout, y justo después se ejecutaba otra vez. Este comportamiento se tuvo en cuenta en la integración con Geofence.

Implementación con Geofence

Esta implementación tiene el objetivo de ejecutar el servicio de Geofence en primer plano, y eso quiere decir que no contiene ninguna restricción de ubicación.

Una vez ya programado, se hizo una primera toma en contacto saliendo a la calle y ver que pasaba. Después de estar durante 10 minutos andando, era increíble como el servicio Geofonce era capaz de notificar todas las transiciones que se producían durante el recorrido.

A continuación, se encuentra un recorrido con esta implementación:

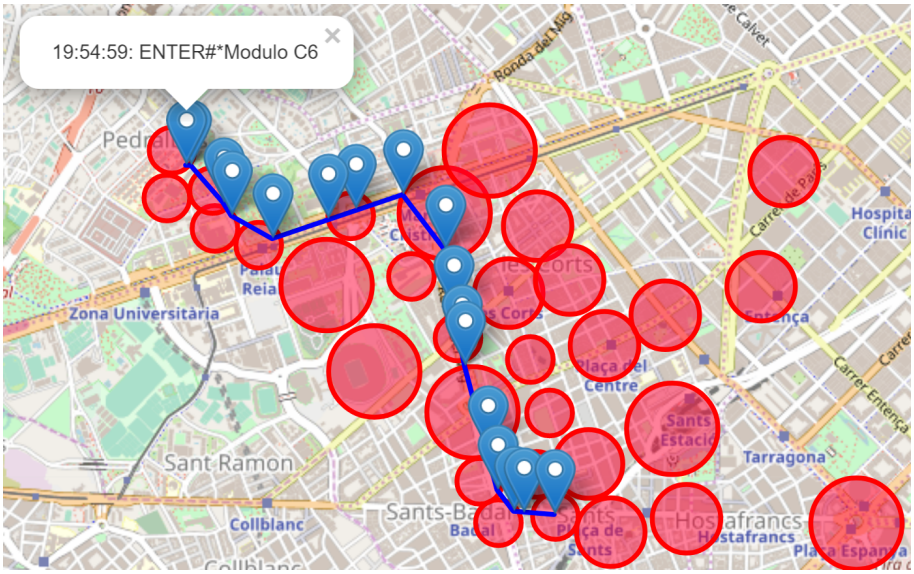


Figura 7.5: Resultados de la implementación en primer plano.

Para la correcta visualización de las notificaciones, se muestra la hora de la notificación, el tipo de transición y las Geofences que lo han producido, haciendo click en cualquier marca del mapa, tal y como se puede ver en la imagen anterior.

Además, vemos como todas las Geofences del recorrido han sido detectadas por el servicio de Geofence, haciendo que el camino formado entre cada una de estas notificaciones, sea muy parecido al camino mostrado en la imagen 7.4.

Una vez implementado y viendo que funcionaba perfectamente, se procedió a la integración de los dos servicios.

Integración de los dos servicios

Una vez que ya funcionaban los dos servicios en sus respectivas aplicaciones, era el momento de desarrollar la integración para que Geofence pudiera ejecutarse en segundo plano.

Esta etapa ha sido la más complicada de todo el desarrollo del proyecto, ya que se estuvo aproximadamente unas 3 semanas intentando que funcionara, ya que desde el principio de la integración solamente se notificaba la primera transición de Geofence.

Después de numerosos cambios en los valores de las configuraciones de las diferentes APIs, tanto de Fused Location como Geofence, no se apreció ningún cambio en el comportamiento, el servicio en segundo plano pasaba a ser olvidado por el teléfono Android y solamente notificaba la primera Geofence. Esto se comprobó con diferentes teléfonos móviles y se vio que el comportamiento era exactamente igual.

Después de muchas pruebas, vimos que en los teléfonos móviles que utilizábamos tenían puesta una restricción de la ubicación en segundo plano, y cuando la desactivábamos funcionaba correctamente, no tan bien como el servicio en primer plano, pero obteníamos notificaciones.

Entonces, después de algunas reuniones, se decidió continuar el proyecto desactivando la restricción en segundo plano, ya que lo que realmente se quería estudiar era el comportamiento de Geofence.

Después de quitar esta restricción, se obtenía un mapa como el de a continuación:

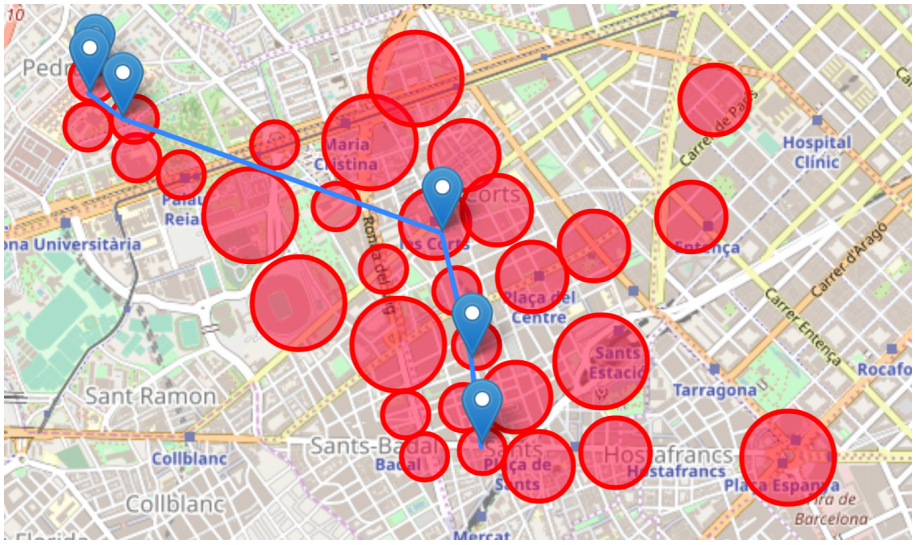


Figura 7.6: Resultados de la implementación en primer plano.

Comparando la imagen anterior con la imagen 7.5, vemos como en esta la tasa de notificaciones es mucho menor haciendo el mismo recorrido.

$$\frac{17 notificaciones foreground}{6 notificaicones background} = 2{,}83$$

El sistema Geofence ejecutado como un servicio en primer plano se obtiene aproximadamente el triple de notificaciones comparado con un servicio en segundo plano. Este calculo es poco significativo por el momento, ya que no son realmente los mismos caminos, y han podido haber factores que ha hecho disminuir el número. Pero después de numerosas ejecuciones de pruebas de implementación, se ha visto que que las notificaciones en segundo plano son menores en número.

7.4.3. Definición de los experimentos

Para proceder a la siguiente sección de análisis de datos, es conveniente que antes se definan una serie de experimentos para poder caracterizarlos:

Definición de la trayectoria

Para poder hacer los diferentes experimentos, antes se debe definir una trayectoria a seguir para saber donde situar las diferentes las Geofences estáticas y así forzar las difernetes notificaciones.

Figura 7.7: Recorrido para los diferentes experimentos.

La ruta que se presenta consta de un total de 7.06 Km, 3km se hacen mediante el tranvía que recorre toda la Avenida Diagonal de Barcelona, y los 4.06 km restantes se hacen andando. Todo esto se efectúa con un tiempo aproximado de 40 minutos por cada recorrido.

Parámetros a estudiar

Se han definido un total de 10 experimentos, los cuales se diferencian en la configuración de algunos parámetros tanto del teléfono móvil como de las diferentes APIs.

A continuación se muestra la lista de los parámetros que definen cada uno de los experimentos:

Radio(m)	**Batería**	**Servicio**	**Modo GPS**	**Aplicación auxiliar**
50	Alta (+15 %)	Background	Alta Precisión	No
100	Baja (-15 %)	Foreground	Baja Precisión	Si
200	-	-	Desactivado	-

Cuadro 7.1: Dedicación en horas por cada tarea.

El parámetro .ªplicación auxiliar"quiere decir que mientras va ejecutándose el servicio Geofence en segundo plano, está presente en primer plano la aplicación Google Maps solicitando datos de ubicación al teléfono móvil.

Al no poder probar cada todas las combinaciones posibles por falta de tiempo, se decide definir una configuración por defecto y a partir de allí ir cambiando cada una de las configuraciones.

Clasificación de los experimentos

La elección del radio de las Geofences en la configuración por defecto viene dado de la recomendación que hace Google. Se encuentra mejor explicado en el apartado *Estado del arte*. Los demás parámetros son elegidos por la configuración por defecto que ta da el dispositivo móvil, ya que, por ejemplo, cuando enciendes el GPS se pone automáticamente el modo de alta precisión.

A continuación se presentan los 3 mapas dado el radio de la Geofence.

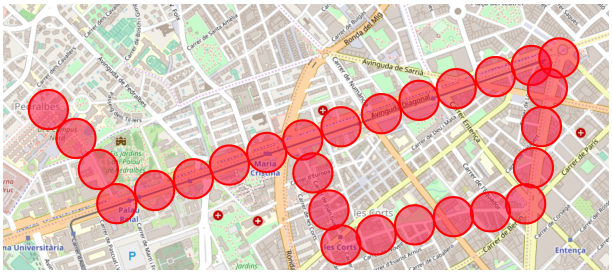


Figura 7.8: Recorrido con Geofences de radio 100 metros.

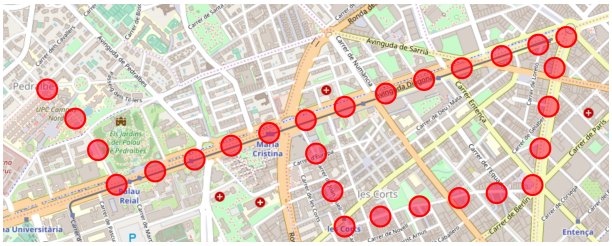


Figura 7.9: Recorrido con Geofences de radio 50 metros.

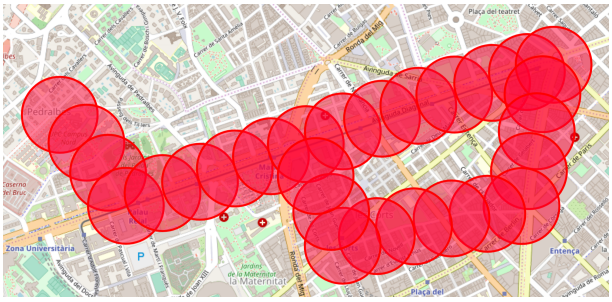


Figura 7.10: Recorrido con Geofences de radio 200 metros.

Definición de los parámetros configurables

Vistos los 3 mapas anteriores, ya podemos empezar a definir los diferentes parámetros definidos en la tabla 7.13, que caracterizan cada unos de los experimentos. Destacar el experimento 1 como la configuración por defecto:

Experimento	**Radio(m)**	**Batería**	**Servicio**	**GPS**	**Aplicación**
1	100	Alta (+15 %)	Background	Alta	No
2	50	Alta (+15 %)	Background	Alta	Si
3	200	Alta (+15 %)	Background	Alta	No
4	100	Alta (+15 %)	Background	Alta	Si
5	100	Baja (-15 %)	Background	Alta	No
6	100	Alta (+15 %)	Background	Baja	No
7	100	Alta (+15 %)	Background	Alta	No
8	50	Alta (+15 %)	Foreground	Alta	No
9	100	Alta (+15 %)	Foreground	Alta	No
10	200	Alta (+15 %)	Foreground	Alta	No

Cuadro 7.2: Parámetros que definen cada uno de los experimentos.

La diferencia que contiene el experimento 7 es por empezar a andar transcurridas 2 horas de la puesta en marcha de la aplicación. Por otro lado, al no poder crear el servicio de Geofence cuando el GPS está desactivado, no se ha podido contemplar ese parámetro como un juego de pruebas.

Definición de los parámetros persistentes

A continuación se detallarán cada uno de los parámetros que son permanentes durante las ejecuciones de cada uno de los experimentos definidos anteriormente, con la excepción de los que se ejecutan en primer plano, ya que se deben cambiar algunos de sus parámetros.

Destacar que en cada uno de los parámetros que se describen a continuación, se deben asignar tiempos en milisegundos. Además, la diferenciación entre alguno de los parámetros si es o no primer plano, se debe a lo que Google te recomienda en cada caso. Cada parámetro se describirá con un pequeño resumen, para recordar lo que significan.

- LocationRequest: Establece la calidad del servicio de las actualizaciones de ubicación.
  - Intervalo:

    Establece la velocidad en milisegundos en el que la aplicación recibe una actualización de ubicación.

- Primer plano. Valor establecido a 5ms.
- Segundo plano. Valor establecido a 3600000ms, que corresponde a 60 minutos

- Fastest interval:

  Establece la velocidad más rápida en milisegundos en que la aplicación puede obtener las actualizaciones de ubicación.

  - Primer plano. Valor establecido a 1ms.
  - Segundo plano. Valor establecido a 300000ms, que corresponde a 5 minutos.

- Prioridad:

  Establece la prioridad de la solicitud de ubicación según como queramos la exactitud de la ubicación.

  - Primer plano. Valor establecido a PRIORITY_HIGH_ACCURACY . Solicita la ubicación más precisa posible.
  - Segundo plano. Valor establecido a PRIORITY_BALANCED_POWER_ACCURACY. Solicita la ubicación con una precisión de aproximadamente 100 metros.

- GeofenceBuilder:

  - Notification responsiveness. Valor establecido a 5ms. La capacidad de respuesta de notificación
  - Loitering delay. Valor establecido a 300000ms, que corresponde a 5 minutos. Tiempo transcurrido entre un ENTER y un DWELL, es decir, establece un tiempo de espera dentro de una Geofence de 5 minutos.

- JobScheduler:

  - Job id = 1. El identificador del Job que se ejecuta en segundo plano.

Una vez definidos cada uno de los experimentos y sus parámetros de configuración, es el momento de salir a la calle y obtener los diferentes datos para poder analizarlos y sacar una información útil.

7.4.4. Desarrollo de los experimentos

Para realizar los diferentes experimentos correctamente, se han decidido utilizar dos teléfonos móviles de características similares, con el objetivo de reducir los impactos de posibles factores externos que no podemos controlar, como por ejemplo, el de otras aplicaciones ejecutándose en segundo plano, o que se apague

el móvil durante el experimento o que por un motivo u otro la aplicación no funcione.

A continuación, se lista los dos móviles utilizados para ejecutar los diferentes experimentos:

- **Móvil Xiaomi Redmi Note 5 personal.** Porcesador Snapdragon 636, 3 GB de RAM y Batería de 4000mAh.
- **Huawei P20 lite del grupo de investigación.** Procesador Kirin 659 a 2,36GHz, 4 GB de RAM y batería de 3000mAh.

7.4.5. Resultados de los experimentos

Una vez acabados los diversos experimentos, es el momento de presentar los resultados obtenidos de cada uno de ellos. Para facilitar la caracterización de los datos, se va a separar la trayectoria en 3 etapas distintas:

- Etapa 1: Agrupa la parte que de la línea recta superior del camino, que corresponde a la calle Avenida Diagonal de Barcelona. Esta parte del camino se realiza en tranvía.
- Etapa 2: Esta etapa se realiza andando, y reúno toda la parte inferior del mapa, hasta llegar otra vez a la linea recta superior.
- Etapa 3: Etapa final del trayecto. Se camina por Geofences definidas que han sido cruzadas anteriormente en la etapa 1.

Para cada experimento se mostrará:

- El tiempo medio de cada etapa.
- El número total de notificaciones producidas en cada etapa.
- Uno o das mapas de los resultados, según las diferencias anteriores.

Después de realizar todos los experimentos, solamente se han podido ver diferencias significativas en el experimento 1, y es por eso que, únicamente ese experimento contiene dos mapas como resultado.

Experimento 1

Este experimento se define como la configuración por defecto de la implementación software realizada. A continuación se definen los parámetros que caracterizan esta configuración:

Primeramente se mostrarán los resultados del teléfono Huawei, y seguidamente, los del teléfono Xiaomi:

Radio(m)	Batería	Servicio	GPS	Aplicación
100	Alta (+15 %)	Background	Alta	No

Cuadro 7.3: Parámetros que definen el experimento 1.

- Resultado Huawei

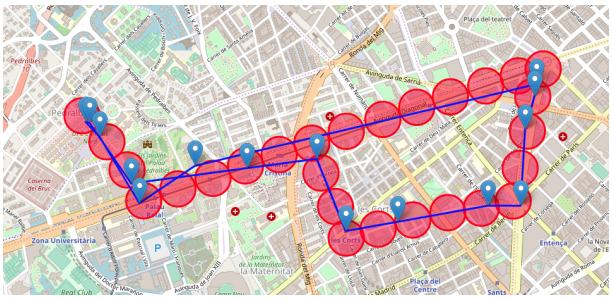


Figura 7.11: Resultado experimento 1 Huawei.

Podemos sacar la siguiente información:

- Número de notificaciones: 18
  - Etapa 1: 6 notificaciones.
  - Etapa 2: 8 notificaciones.
  - Etapa 3: 4 notificaciones.
- Tiempo medio entre notificaciones: 4.67 minutos
  - Etapa 1: 5.2 minutos.
  - Etapa 2: 3.99 minutos.
  - Etapa 3: 4.7 minutos.

- Resultado Xiaomi

Figura 7.12: Resultado experimento 1 Xiaomi.

- Número de notificaciones: 38.
  - Etapa 1: 19 notificaciones.
  - Etapa 2: 11 notificaciones.
  - Etapa 3: 8 notificaciones.
- Tiempo medio entre notificaciones: 4.7 minutos
  - Etapa 1: 3.6 minutos.
  - Etapa 2: 5.2 minutos.
  - Etapa 3: 5.3 minutos.

Comentarios

Al ver los resultados de este primer experimento, parece que el comportamiento de Geofence depende mucho del teléfono móvil, ya que se pueden apreciar numerosas diferencias entre las dos ejecuciones en cuanto al número de notificaciones. Por otra parte, el tiempo medio es bastante parecido, ya que ronda unos 4.7 minutos en media.

Pensando el motivo de esta diferencia en la cantidad de notificaciones, se concluye que lo más probable es que en el móvil personal, antes de empezar el experimento, se ejecutara algún tipo de servicio en segundo plano no restringido que pidiera ubicación, haciendo que Geofence cogiera esa información para notificar más.

Se llega a dicha conclusión por el motivo de que en los siguientes experimentos, no se aprecia una diferencia significativa entre las dos ejecuciones, y es por eso que solamente se ha proporcionado una imagen del mapa resultante.

Por otra parte, el motivo de que el tiempo sea prácticamente igual es porque las diferentes marcas del mapa de Xiaomi contiene más de 1 notificación a la vez, haciendo que aumente el número de notificaciones pero no reduzca el tiempo entre ellas.

Experimento 2

Este segundo experimento se diferencia por definir las Geofences con un radio de 50 metros, es decir, el doble de pequeñas que las Geofences por defecto. A continuación se muestran los parámetros de configuración:

Radio(m)	**Batería**	**Servicio**	**GPS**	**Aplicación**
50	Alta (+15 %)	Background	Alta	Si

Cuadro 7.4: Parámetros que definen el experimento 2.

- Resultado

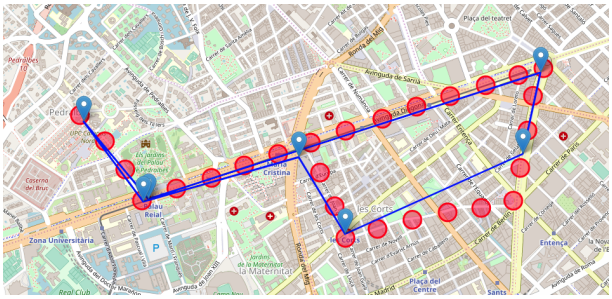


Figura 7.13: Resultado experimento 2.

- Número de notificaciones: 12.
  - Etapa 1: 5 notificaciones.
  - Etapa 2: 2 notificaciones.
  - Etapa 3: 5 notificaciones.
- Tiempo medio entre notificaciones: 11.13 minutos
  - Etapa 1: 10.3 minutos.
  - Etapa 2: 11.8 minutos.
  - Etapa 3: 11.3 minutos.

Comentarios

Viendo los resultados anteriores podemos ver lo sorprendente que es el tiempo entre notificaciones. Sube 6 minutos respecto al experimento 1, además de bajar considerablemente el número de notificaciones.

Con la ejecución de este experimento se pueden apreciar dos indicios de comportamiento de Geofence:

- La notificación ENTER solamente se produce siempre y cuando el sistema Geofence despierta dentro de una Geofence definida. Esto hace que el tiempo entre notificaciones se dispare y que las notificaciones bajen respecto el experimento anterior.
- Notifica un EXIT aproximadamente 5 minutos después de un ENTER. Con esto podemos ver que Geofence despierta cada 5 minutos, algo que también se produce en el experimento anterior.

Experimento 3

Finalmente, en cuanto al estudio de radios de Geofence en estado de segundo plano, se decide añadir una última configuración de 200 metros. El objetivo de este experimento es ver cómo notifica Geofence una vez que los círculos se superponen unos a los otros.

A continuación se muestran los parametros de configuración para este experimento:

Radio(m)	**Batería**	**Servicio**	**GPS**	**Aplicación**
200	Alta (+15 %)	Background	Alta	No

Cuadro 7.5: Parámetros que definen el experimento 3.

- Resultado

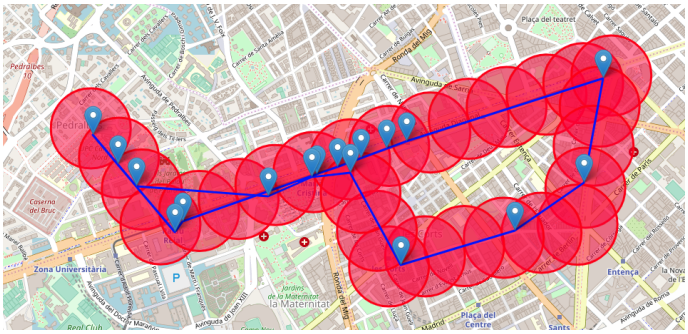


Figura 7.14: Resultado experimento 3.

- Número de notificaciones: 30.
  - Etapa 1: 14 notificaciones.
  - Etapa 2: 6 notificaciones.
  - Etapa 3: 10 notificaciones.
- Tiempo medio entre notificaciones: 5.437 minutos
  - Etapa 1: 5.3 minutos.
  - Etapa 2: 5.12 minutos.
  - Etapa 3: 5.89 minutos.

- Comentarios

Una vez visto el resultado, se puede apreciar que el camino formado por las notificaciones es bastante semejante, pero en este experimento se ve como el número de notificaciones ha aumentado casi el doble. Esto se debe a que en un mismo instante de tiempo se notifica con más de una notificación. Además, se tienen que sumar algunas notificaciones de DWEL, ya que hay algunas Geofences que permaneces más de 5 minutos por ser tan grandes.

En conclusión, aumentar los radios de Geofence de 100 a 200 metros, donde solo obtienes superposición de Geofence, no te aporta nada, ya que con las de 100 metros eres capaz de dibujar el mismo camino disminuyendo a casi a la mitad el número de notificaciones.

Experimento 4

Después de haber estudiado diversas combinaciones de radio de Geofence, se ha decidido analizar cómo puede afectar una aplicación en primer plano, haciendo peticiones de ubicación, al servicio de Geofence ejecutándose en segundo plano.

Radio(m)	Batería	Servicio	GPS	Aplicación
100	Alta (-15 %)	Background	Alta	Si

Cuadro 7.6: Parámetros que definen el experimento 4.

- Resultado

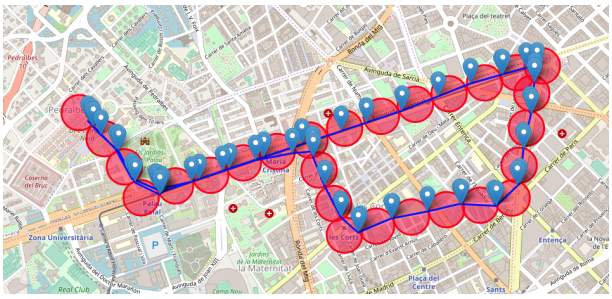


Figura 7.15: Resultado experimento 4.

- Número de notificaciones: 70.
  - Etapa 1: 28 notificaciones.
  - Etapa 2: 24 notificaciones.
  - Etapa 3: 18 notificaciones.
- Tiempo medio entre notificaciones: 1.8 minutos
  - Etapa 1: 0.81 minutos.
  - Etapa 2: 1.1 minutos.
  - Etapa 3: 1.05 minutos

- Comentarios

Viendo el resultado de este experimento en la imagen del mapa, se puede contemplar como todas las Geofences definidas han sido notificadas por el servicio de ubicación desarrollado. Viendo esto, podemos concluir que:

- Cuando hay una aplicación en primer plano solicitando actualizaciones de ubicación, Geofence es capaz de "despertarse"para obtener dichas actualizaciones, y así, notificar que se ha producido una notificación.
- Al ver que se ha producido transición en cada una de las Geofences definidas, podemos decir que el momento en que al servicio Geofence le llega una actualización de ubicación, éste solamente notifica cuando se ha producido una transición, es decir, dada una latitud y longitud en un instante de tiempo:
  - Primeramente comprueba si se debe enviar un EXIT de alguno de los ENTERs anteriores a ese instante de tiempo, si es así, debes comprobar si ha salido de esa Geofence. En ese caso deberás notificar un EXIT.
  - Después, mira si la nueva ubicación está dentro de una Geofenceo, si es así, notifica con un ENTER, sino no hace nada.

Experimento 5

Ha habido situaciones en las que el teléfono móvil se quedaba sin batería, y durante las reuniones semanales nos preguntábamos qué pasa con el servicio de Geofence cuando la batería es menor a 15.

A continuación se muestran la configuración cuando el teléfono móvil se encuentra con batería baja:

Radio(m)	**Batería**	**Servicio**	**GPS**	**Aplicación**
100	Baja (-15 %)	Background	Alta	No

Cuadro 7.7: Parámetros que definen el experimento 5.

- Resultado

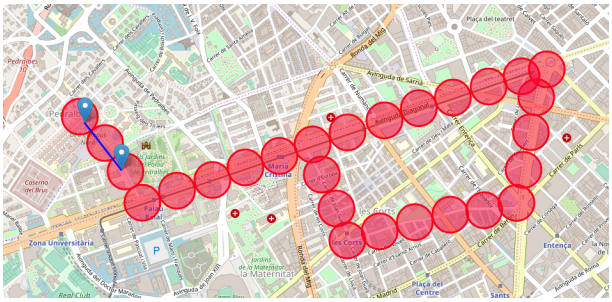


Figura 7.16: Resultado experimento 5.

- Número de notificaciones: 4.
  - Etapa 1: 4 notificaciones.
  - Etapa 2: 0 notificaciones.
  - Etapa 3: 0 notificaciones.
- Tiempo medio entre notificaciones: 5.20 minutos
  - Etapa 1: 5.20 minutos.
  - Etapa 2: - minutos.
  - Etapa 3: - minutos

- Comentarios

Viendo el resultado anterior, vemos como solamente se han notificado 4 transiciones, y comprendiendo que solamente hemos cambiado el parámetro de la batería, se puede llegar a la siguiente conclusión, ya que a los dos teléfonos móviles les ha pasado lo mismo:

- Parece que el móvil pasa a un estado bastante restringido en cuanto a los servicios de segundo plano.

Experimento 6

En este experimento se prueba el parámetro de modo ahorro de GPS, el cual utiliza WiFi y redes móviles para determinar la ubicación del dispositivo. A continuación se muestran los parámetros que caracterizan este experimento:

Radio(m)	Batería	Servicio	GPS	Aplicación
100	Alta (-15 %)	Background	Baja	No

Cuadro 7.8: Parámetros que definen el experimento 6.

- Resultado

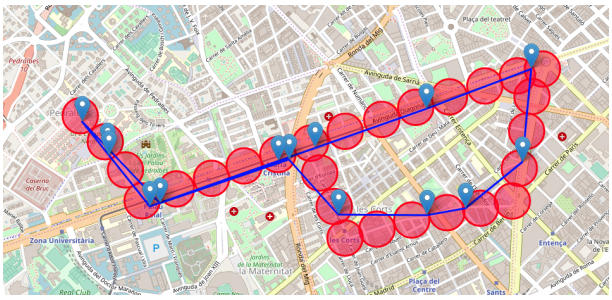


Figura 7.17: Resultado experimento 6.

- Número de notificaciones: 28.
  - Etapa 1: 14 notificaciones.
  - Etapa 2: 8 notificaciones.
  - Etapa 3: 6 notificaciones.
- Tiempo medio entre notificaciones: 5.63 minutos
  - Etapa 1: 5.20 minutos.
  - Etapa 2: 5.8 minutos.
  - Etapa 3: 5.9 minutos

- Comentarios

Viendo el resultado del mapa anterior, se puede apreciar que estableciendo el GPS cn el modo de ahorro de batería, el comportamiento en las notificaciones de Geofence es significativamente parecido a los anteriores.

Lo único que podemos comentar en esta sección es que el número de notificaciones es considerablemente normal, 28 notificaciones, algo que no concuerda con el número de avisos del mapa, ya que solamente se muestran 14. Eso es

porque en cada instante de tiempo de aviso de Geofence, este notifica 2 transacciones a la vez en todos los casos, uno del EXIT de la Geofence anterior, y el otro del ENTER de la nueva Geofence.

Con esto quiero llegar a la conclusión que cambiando el GPS en modo ahorro, hace bajar el número de avisos de la aplicación, pero no el número de notificaciones.

Experimento 7

Para poder estudiar hasta qué punto el sistema operativo olvida tanto el servicio Geofence como el servicio en segundo plano, se ha diseñado este experimento con los mismo parámetros por defecto, pero ejecutándose 2 horas antes de empezar a caminar.

Radio(m)	**Batería**	**Servicio**	**GPS**	**Aplicación**
100	Alta (-15 %)	Background	Alta	No

Cuadro 7.9: Parámetros que definen el experimento 7.

- Resultado

Para el resultado de este experimento no es posible mostrar el mapa resultante, ya que solamente contamos con dos notificaciones, una de ENTER de la puesta en marcha de la aplicación, y la otra, 5 minutos después, de un DWELL por permanecer dentro de la Geofence más de 5 minutos.

Al salir a hacer la prueba, después de esperar 2 horas, el sistema en segundo plano o Geofence es olvidado por el sistema operativo.

Experimento 8

Una vez acabados todos los experimentos establecidos en el servicio de segundo plano, es hora de empezar con los experimentos de primer plano, los cuales solo se diferencian en el tamaño del radio de la Geofence.

A continuación se muestran los parámetros de configuración del experimento 8:

Radio(m)	**Batería**	**Servicio**	**GPS**	**Aplicación**
50	Alta (-15 %)	Foreground	Alta	No

Cuadro 7.10: Parámetros que definen el experimento 8.

- Resultado

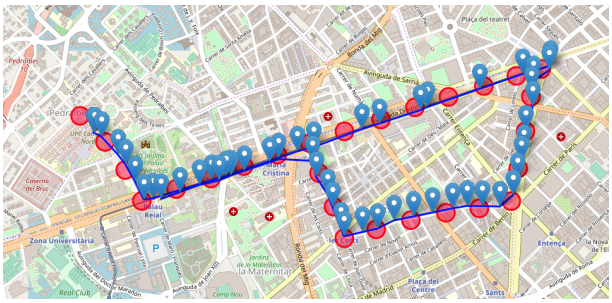


Figura 7.18: Resultado experimento 8.

- Número de notificaciones: 57.
  - Etapa 1: 23 notificaciones.
  - Etapa 2: 24 notificaciones.
  - Etapa 3: 10 notificaciones.
- Tiempo medio entre notificaciones: 1.17 minutos
  - Etapa 1: 0.9 minutos.
  - Etapa 2: 1.28 minutos.
  - Etapa 3: 1.32 minutos

- Comentarios

Se puede apreciar como la mayoría de Geofece han sido reportadas como transición, teniendo un número de notificaciones de 57. Además, se puede observar como las transiciones de EXIT se obtienen por separado del ENTER siguiente, ya que a simple vista vemos que tenemos notificaciones fuera de las áreas de las Geofences, y que los ENTERS no se notifican a la vez que los EXITS, por la numerosa cantidad de 'Points'.

Gracias a esto, podemos deducir que el comentario hecho del experimento 4 podría ser bastante acertado. Solamente despierta la notificación cuando Geofence hace sus cálculos internos de cuándo hacer una notificación o no.

Experimento 9

Este experimento se podría decir que es la configuración por defecto del servicio de Geofence en primer plano. Su configuración se muestra a continuación:

Radio(m)	Batería	Servicio	GPS	Aplicación
100	Alta (-15 %)	Foreground	Alta	No

Cuadro 7.11: Parámetros que definen el experimento 9.

- Resultado

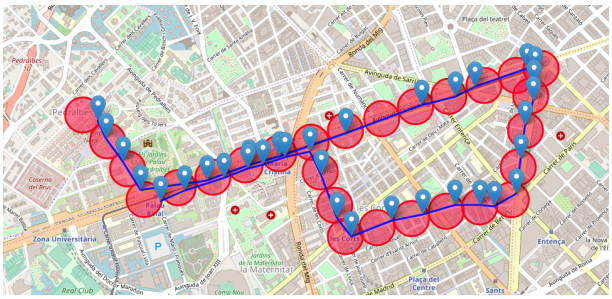


Figura 7.19: Resultado experimento 9.

- Número de notificaciones: 60.
  - Etapa 1: 23 notificaciones.
  - Etapa 2: 25 notificaciones.
  - Etapa 3: 12 notificaciones.
- Tiempo medio entre notificaciones: 1.052 minutos
  - Etapa 1: 0.92 minutos.
  - Etapa 2: 1.08 minutos.
  - Etapa 3: 1.15 minutos

- Comentarios
El resultado obtenido en este experimento es bastante parecido al anterior, encontrando la única diferencia de que el número de avisos pintados en el mapa han bajado en número. Esto pasa porque el EXIT de la Geofence anterior se notifica con un ENTER de la nueva área de Geofence.

Igualmente se obtiene la misma información y mismo recorrido.

Experimento 10

Para concluir los resultados de los experimentos se muestra la misma configuración que la anterior, pero con el radio de las Geofences definidas a 200 metros.

Radio(m)	**Batería**	**Servicio**	**GPS**	**Aplicación**
200	Alta (-15 %)	Foreground	Alta	No

Cuadro 7.12: Parámetros que definen el experimento 10.

- Resultado

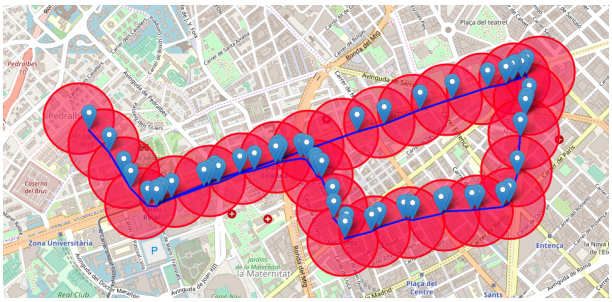


Figura 7.20: Resultado experimento 10.

- Número de notificaciones: 74.
  - Etapa 1: 31 notificaciones.
  - Etapa 2: 30 notificaciones.
  - Etapa 3: 13 notificaciones.
- Tiempo medio entre notificaciones: 1.127 minutos
  - Etapa 1: 1.09 minutos.

- Etapa 2: 1.51 minutos.
- Etapa 3: 1.21 minutos

- Comentarios

Viendo el mapa de la figura anterior, se puede apreciar similitud con el experimento 8, donde los radios eran de 50 metros. Cabe destacar que los EXITS son notificados en el mismo instante de tiempo, y lo que hace que haya más avisos son las transacciones de DWELL, ya que estás más de 5 minutos dentro de las distintas Geofence.

Igualmente, gracias a las diversas notificaciones podemos dibujar el camino hecho perfectamente.

7.5. Análisis de los resultados

En cada uno de los experimentos se han obtenido dos resultados, por el motivo de la utilización dos teléfonos móviles. Para reducir la información redundante de cada experimento, solamente se expondrán los dos resultados si se manifiesta alguna diferencia entre las dos ejecuciones.

7.5.1. Primeros análisis del sistema Geofence

Durante las ejecuciones de prueba de las diferentes implementaciones que contenían el servicio de Geofences se encontraron los siguientes comportamientos:

- En la puesta en marcha de la aplicación, Geofence te notifica con una primera transición de EXIT (salida) de todas las Geofences definidas en el sistema. Pero en el caso de que inicies la aplicación y te encuentres dentro de una o más Geofences, la primera notificación que te llega es de un ENTER de esas Geofences, y seguidamente te llega otra notificación EXIT con todas las Geofences, las cuales no te ubicas dentro. Entonces al iniciar la aplicación te encuentras con estas situaciones:
  - Primera notificación de ENTER, en caso de que te encuentres dentro de alguna Geofence.
  - Las Geofences que están definidas en el sistema y que en el momento de iniciar la aplicación no te encuentras dentro de ellas, son mostradas en la aplicación con una transición EXIT.
- En las ejecuciones en el servicio de segundo plano, el tiempo entre cada una de las notificaciones de transición de Geofence llega en media a 5 minutos.

- Numerosas veces un EXIT de una Geofence viene acompañado de un ENTER de otra. Las primeras impresiones son que necesitamos un ENTER para obtener un EXIT de una Geofence. Esto es algo que en el siguiente capítulo se desmiente.
- En cuanto al servicio de segundo plano sin las actualizaciones de ubicación, la duración de la ejecución es de 10 minutos, recibiendo un timeout, y justo después se ejecuta otra vez.

7.5.2. Análisis de los problemas surgidos durante la etapa de correción de errores

A continuación se detallarán estos problemas y sus respectivas soluciones:

- **Implementación del servicio en background**

  Durante las primeras fases de búsqueda de información sobre los servicios en background y de análisis de los diferentes modos de implementación del modelo, nos dimos cuenta que no sería una tarea sencilla de hacer.

  Como se detalla en la descripción de la tarea de *diseño e implementación* de la planificación inicial, su última fase era la integración de la tecnología Geofence con la implementación del servicio en background. Para probar su correcto funcionamiento era necesario salir a la calle y ver si realmente funcionaba con el móvil inactivo y vimos que solamente funcionaba con la primera notificación de Geofence.

  Después de mucho análisis de código y búsqueda de información, nos dimos cuenta que el problema que teníamos era sobre la restricción que tiene Android sobre los dispositivos de versión 8 o superior. Resumiendo, esta restricción mata todos los procesos en background que utilicen localización con el objetivo de aumentar la el tiempo de vida de la batería del dispositivo.

  Dicho esto, después discutirlo dentro de las reuniones semanales, tomamos la decisión de inhabilitar dicha restricción para analizar el verdadero comportamiento de esta tecnología, ya que es el principal objetivo de este proyecto.

- **Definición de los diferentes experimentos**

  Una vez que veíamos que la implementación software funcionaba correctamente, era el momento de definir diferentes experimentos para ver su comportamiento en diferentes entornos de ejecución.

  Durante una de las reuniones semanales, se presentaron los diferentes parámetros de configuración de la aplicación, y vimos que eran demasiados para probar todas sus combinaciones. Así que decidimos definir una configuración por defecto, y a partir de ahí ir cambiando cada parámetro

de configuración. Cogiendo esta solución, nos hicimos que saliera un total de 10 experimentos diferentes a ejecutar.

7.5.3. Conclusiones de los experimentos

Recopilación de los resultados

Para poder empezar a dar una visión general del comportamiento de Geofence, se muestra a continuación una tabla con cada uno de los datos más importantes recogidos. Cabe destacar que en esta tabla se ha añadido la variación de la descarga de batería del teléfono móvil producida en cada uno de los experimentos.

Para poder poner todo en una misma columna, se ha reducido el significado de cada columna:

- Exp: corresponde al número de experimento.
- 1..3: el número que define la etapa.
- Batería: variación de descarga de la batería.
- Primer grupo corresponde a las notificaciones y el segundo al tiempo medio entre avisos en minutos.

Exp	**1**	**2**	**3**	**Total**	**1**	**2**	**3**	**Total**	**Batería**
1	19	11	8	38	3.6	5.2	5.3	4.7	4
2	5	2	5	12	10.3	11.8	11.3	11.13	1
3	14	6	10	30	5.3	5.12	5.89	5.43	3
4	28	14	18	70	0.81	1.1	1.05	1.8	12
5	4	0	0	4	5.20	-	-	5.20	1
6	14	8	6	28	5.20	5.8	5.9	5.63	3
7	0	0	0	-	-	-	-	-	x
8	23	24	10	57	0.9	1.28	0.9	1.17	20
9	23	25	12	60	0.92	1.08	1.15	1.06	17
10	31	30	13	74	1.09	1.51	1.21	1.12	15

Cuadro 7.13: Resumen de los resultados.

-Comentario del consumo de batería

Viendo cada una de las variaciones en la descarga de batería de la tabla anterior, se ve claramente la diferencia entre los dos modos de ejecución. En

el servicio de segundo plano se obtiene una descarga máxima de 4 puntos, sin contar el experimento 4, el cual había una aplicación externa ejecutándose en primer plano. Con esto quiero decir que, si inicialmente tenia un 90 % de batería, al acabar el respectivo experimento tenía 86 %, casi insignificante, y teniendo en cuenta también la existencia de otras servicios ejecutándose en segundo plano.

Por otra parte, vemos que la variación máxima de batería en el primer plano llega a los 20 puntos, algo normal, ya que la aplicación ha estado consumiendo muchos recursos del teléfono. El consumo de la pantalla o de Internet podrían ser los recursos que más hayan perjudicado a esta variación de energía consumida.

Siendo sinceros, este apartado solamente nos da una visión de cómo realmente consume un teléfono cuando se encuentra con la pantalla encendida o apagada. En nuestro caso, no es posible saber lo que realmente consume en sí el sistema Geofence, ya que existen numerosos factores que afectan a la batería del teléfono móvil.

Conclusiones del comportamiento de Geofence

Los siguiente puntos son las diferentes conclusiones que se han extraído a partir de la ejecución de todos los experimentos.

- Aunque llegue un timeout al servicio en segundo plano, este se despierta justo unos minutos después, sin afectar al comportamiento de Geofence.
- Existencia de una primera notificación en el momento de arrancar el servicio de Geofence, conteniendo una lista de todas las Geofences definidas en la implementación software con una notificación de EXIT. Si se da el caso de que en ese momento ya estás dentro de una, dicha Geofence no se encuentra en la lista anterior, y será notificada con un ENTER.
- El incremento o descenso de los radios de Geofence no aporta información adicional, solamente obtiene más avisos:
  - Para el incremento, obtienes transiciones de DWELL, ya que tardas algo más de 5 minutos en cruzar una Geofence.
  - Para la disminución, obtienes los EXITs separados de los ENTERs siguientes. Esto pasa porque cuando Geofence se da cuenta de que has salido de una área, en ese momento no te encuentra dentro de ninguna Geofence.
- El tiempo entre avisos de Geofences se encuentra de media en poco más de 5 minutos. Siempre y cuando no tengas un factor externo que te solicite ubicación. Es el caso de utilizar una aplicación como Google Maps[2] en primer plano, en ese caso el tiempo medio es de 1 minuto.

[2]https://www.google.com/maps

- Cuando a Geofence le llega una actualización de ubicación, comprueba que Geofences tiene guardadas como un ENTER sin aún notificar un EXIT. Si dicha ubicación no esta fuera del radio de esas Geofence, no avisa a la aplicación, si es el caso de que está fuera, despierta a la aplicación notificando un EXIT. Además, si esta nueva ubicación se encuentra dentro de otras Geofence, se notificará juntamente con los EXITs anteriores.
- Estableciendo el GPS en modo batería, no afecta al comportamiento de Geofence, cuando este se ejecuta en segundo plano.
- Con batería baja, Geofence solamente funciona al principio de recorrido. Cuando el servicio de segundo plano recibe el timeout del sistema operativo después de 10 minutos, este no vuelve a despertar.
- Aunque se haya definido Fused Location como proveedor de ubicación dentro del servicio de segundo plano, Geofence solamente puede obtener su ubicación pasados 30 minutos, tal y como está definido en la configuración del proveedor.
- Parece que Geofence tiene definido en su estructura un proveedor de ubicación que solamente proporciona una actualización de ubicación al sistema si:
  - Han pasado 5 minutos de la anterior actualización de ubicación.
  - Otra aplicación se encuentra solicitando actualizaciones de ubicación. Estas actualizaciones son recogidas por Geofence, y este es capaz de ver si hay que notificar o no a la aplicación, dependiendo de si se ha producido una transición.
- En un 95 % de notificaciones el proveedor es Fused Location. Todas las notificaciones restantes corresponden al GPS como proveedor, mostrando así diferentes datos como la velocidad que se iba o altitud en la que te encuentras.
- Después de dos horas de ejecución sin notificar ninguna transición, parece que Geofence o el servicio de segundo plano es olvidado por el sistema operativo.

Capítulo 8

Desarrollo algoritmo rastreo

8.1. Introducción

Una vez conocidos los distintos comportamientos de las Geofences a partir de los análisis de experimentos en el capítulo anterior, es el momento de desarrollar un algoritmo capaz de rastrear un teléfono móvil con Android.

El algoritmo desarrollado en este capítulo, es capaz de rastrear cualquier teléfono, siempre y cuando el usuario haya pulsado un botón de START, o abra la notificación que llega desde los servicios de Firebase de Push Notification.

Cabe recordar que, para utilizar este algoritmo se debe quitar la restricción de ubicación en segundo plano del teléfono móvil, sino la implementación software no será capaz de notificar, haciendo que no funcione correctamente.

8.2. Objetivos

Los principales objetivos para cumplir el desarrollo de esta sección, y así acabar el Treball de Final de Grau, son los siguientes:

- Desarrollo de un algoritmo de rastreo basado en Geofences dinámicas.
    - El algoritmo debe ser capaz de rastrear incluso cuando el dispositivo móvil se encuentre a velocidades moderadas, como por ejemplo, mientras se nos movemos en coche por la ciudad.
- Iniciar el algoritmo de rastreo a través de una notificación proveniente de Firebase Push Notification.

8.3. Recursos

8.3.1. Firebase

Firebase es una combinación de numerosos servicios de Google que incluye mensajería instantánea, autenticación de usuarios, bases de datos en tiempo real, almacenamiento, etc.

Firebase contiene numerosas ventajas, como por ejemplo, la ausencia de mantenimiento de las máquinas, virtualización del estado de cada instancia y la posibilidad de configuración a través de su plataforma web. Aparte de esto, para gestionar cualquier de los servicios anteriores, Firebase contiene una gran cantidad de documentación disponible de forma pública, dónde lo explica con ejemplos y tutoriales visuales.

Firebase contiene planes de precio, pero, el servicio que se utilizará en este proyecto se ofrece de manera totalmente gratuita.

Figura 8.1: Logotipo de Firebase.

En nuestro caso, utilizaremos un servicio para enviar un Push Notification que proporciona Firebase llamado Firebase Cloud Messaging.

Firebase Cloud Messaging

Firebase Cloud Messaging permite notificar a aplicaciones móviles instaladas en un dispositivo en tiempo real. Por ejemplo, puede enviar información sobre una nueva actualización de una aplicación y al pulsar se actualiza sólo o notificaciones para enviar mensajes de publicidad para que compres algún producto, como por ejemplo lo que hacen Amazon[1] y Aliexpress[2].

Con FCM, se puede enviar dos tipos de mensajes a los clientes [43]:

- Mensajes de notificación: Muestran automáticamente la notificación en el dispositivo del usuario destino. En este caso, solamente se recibe la

[1] https://www.amazon.es/
[2] https://es.aliexpress.com/

Figura 8.2: Push notification Firebase Cloud Messaging. Recuperado de:

notificación cuando la aplicación se encuentra en primer plano, y sino, se presenta en la barra de notificaciones del teléfono móvil.

- Mensajes de datos: Te llegan directamente a la aplicación, sin que se despliegue una notificación. Este tipo de mensajes pueden configurarse para que le lleguen directamente al servicio de segundo plano, pero FCM no puede enviar mensajes a aplicaciones donde se encuentre configurado la restricción de segundo plano [44]. Encontramos el mismo problema durante la implementación de Geofence.

Para este proyecto se utilizarán los mensajes de notificación, ya que son los únicos que te Firebase te deja hacer desde su consola web. El motivo de la no utilizar el tipo de mensajes de datos, es por que no se pueden enviar directamente desde la consola web de Firebase, sino que necesitas un servidor que se conecte a partir de un token con la plataforma, y éste envíe el mensaje al dispositivo. Esto es algo que no se tenía como objetivo en el proyecto, y es por eso que al final nos decantamos por Push Notification con mensajes de notificaciones.

8.4. Desarrollo

8.4.1. Definición de la estrategia

Primeramente, para poder desarrollar el algoritmo de rastreo, necesitamos seguir una estrategia basada en los diferentes comportamientos de Geofences explicados en el apartado anterior.

Como hemos visto, el tiempo medio entre notificaciones no llega a unos 5 minutos, eso quiere decir que una persona puede andar 330 metros sin ser notificado por Geofence, suponiendo que la velocidad de una persona son 4km/h. Con otras palabras, necesitamos hasta 3 Geofences de 100 metros de radio en línea recta para que pueda llegar la segunda notificación. Dicho esto suena fácil, pero en el caso de un coche que va a unos 50 km/h por ciudad, puede llegar a recorrer hasta 4.2 kilómetros, es decir, un total de 21 Geofence en línea recta para que salte la notificación.

Con el razonamiento anterior, funcionaría bien si fuéramos siempre a la misma dirección, pero tanto un coche como una persona puede cambiar su dirección en cualquier momento. Es por eso que se propone un algoritmo, el cual dibuje 3

niveles en forma de cuadrado, donde su conjunto forma un Grid. Entonces, definimos como Grid el conjunto de Geofences reparativas en un espacio cuadrado formado por distintos niveles. El nivel del Grid marcará el radio de las Geofences, es decir, el perímetro total del Grid. Cuanto menor sea el nivel, menor será el radio.

En nuestro caso hemos decidido formar 3 niveles de Grid, donde cada nivel está formado por 9 Geofences formando un cuadrado. Dicho cuadrado ocupa el mismo espacio que el nivel de Grid superior. Ver en la imagen 8.9.

Funcionamiento del algoritmo

El proceso en segundo plano, que implementa las Geofences dinámicas, empieza cuando el usuario pulsa la notificación proveniente de Push Notificaction de Firebase. En ese momento, se ejecuta una parte de código que crea un servicio en segundo plano, y hace un petición de ubicación para poder centrar inicialmente todo el Grid.

Una vez centrado el Grid, el servicio Geofence empezará notificando todas las transiciones que pueda. Sin embargo, cuando se dé el caso de que la transición se produzca fuera del primer nivel del Grid, se volverá a centrar todo él en la ubicación dada por dicha notificación.

El objetivo de la presencia de diversos niveles es tener siempre al usuario en el primero, es decir, siempre debe ser rastreable cuando se encuentre en el centro de todo el Grid, y si sale de éste, obtener una notificación del otro nivel para poder mover todos los centros de los niveles del Grid a la localización de la notificación.

Aparte de esto, se ha aprovechado que las Geofences se sobrescriben por nombre. Cada vez que se forma un nuevo Grid, solamente hace falta añadir las nuevas Geofences, ya que son las mismas pero solamente cambiando su latitud y longitud.

8.4.2. Algoritmo de rastreo

A continuación, se muestra cada nivel de Grid y como se calcula todas las ubicaciones de cada Geofences según su nivel, dada la fórmula de Haversine[3]. Además, destacar que cada nivel va encima de otro, es decir, tienen el mismo centro de origen.

Nivel 1 del Grid

Como se ve en la imagen, el nivel 1 es la única configuración que no deja espacios en blanco entre las Geofences, ya que lo que queremos es forzar

[3] https://es.wikipedia.org/wiki/F%C3%B3rmula_del_haversine

notificaciones para saber dónde está el usuario en cada momento.

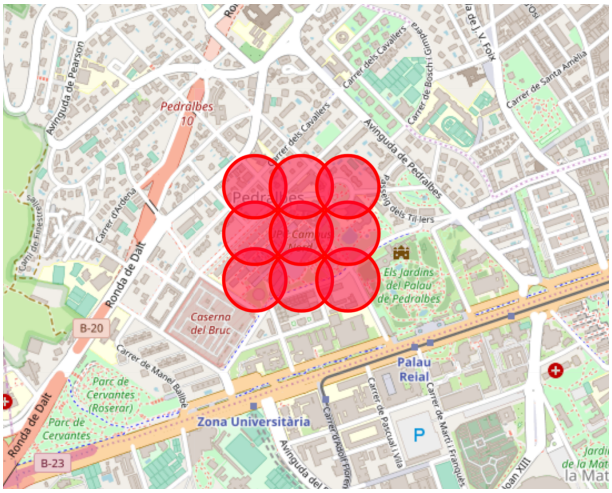


Figura 8.3: Nivel 1 del Grid.

A continuación, se muestra el fragmento de código que calcula cada latitud y longitud de cada Geofence del primer nivel.

```
//offsets in meters
Double dn = 150.0;
Double de = 150.0;

//Coordinate offsets in radians
Double dLat = dn/EARTHRATIO;
Double dLon = de/(EARTHRATIO*Math.cos(Math.PI*ini_lat/180.0));

int x = 0;
int y = 0;
for (int i = -1; i <= 1; ++i){
    for (int j = -1; j <= 1; ++j) {
        double lat = ini_lat + i*dLat * 180/Math.PI;
        double lon = ini_lon + j*dLon * 180/Math.PI;
        this.addGeofence(lat,lon,"Grid" + x + y + '0', 100);
        Log.d("Geofence",lat + "," + lon + ",Grid" + x + y + "0,100");
        y = y + 1;
    }
    y = 0;
    x = x +1;
}
```

Como se aprecia en el fragmento de código anterior, el radio de cada área es de 100 metros, formando un perímetro aproximado de 2400 metros. Si el usuario se encuentra en el centro, situación inicial de la aplicación, tenemos un margen de maniobra de 1 km en cualquier dirección.

Nivel 2 del Grid

La siguiente imagen muestra el segundo nivel del Grid, el cual ahora sí que se aprecian los espacios en blanco entre las diferentes áreas de Geofence. El hueco libre que hay entre las Geofence en principio no afecta al correcto funcionamiento del algoritmo, ya que siempre nos llegará el EXIT del primer nivel o el ENTER de alguna Geofences del segundo nivel.

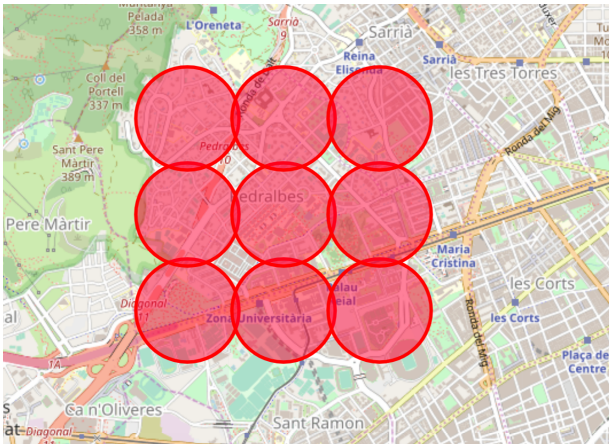


Figura 8.4: Nivel 2 del Grid.

Cuando se da una notificación de transición en este nivel se mueven los centros de cada nivel, a la ubicación dada por la notificación.

A continuación, se muestra el fragmento de código que crea el segundo nivel del Grid a construir.

```
1  //offsets in meters
2  dn = 600.0;
3  de = 600.0;
4  //Coordinate offsets in radians
5  dLat = dn/EARTHRATIO;
6  dLon = de/(EARTHRATIO*Math.cos(Math.PI*ini_lat/180.0));
7
8  x = 0;
9  y = 0;
10 for (int i = -1; i <= 1; ++i){
11     for (int j = -1; j <= 1; ++j) {
12         double lat = ini_lat + i*dLat * 180/Math.PI;
13         double lon = ini_lon + j*dLon * 180/Math.PI;
14         this.addGeofence(lat,lon,"Grid" + x + y + '1', 320);
15         Log.d("Geofence",lat + "," + lon + ",Grid" + x + y + "1,320");
16         y = y + 1;
17     }
18     y = 0;
19     x = x +1;
20 }
```

En este caso, cada una de las Geofences están construidas con un radio de 320 metros. Eso quiere decir que forman un rectángulo de perímetro de 7680 metros. Y pensando en el caso inicial, la aplicación tiene un margen de maniobra de 960 metros en todas las direcciones cuando el usuario se encuentra en el centro del Grid. Entonces, para crear un nuevo Grid, el usuario debe recorrer como mínimo 1km.

Nivel 3 del Grid

En este nivel se aprecia el gran perímetro que contiene, ya que las Geofences están definidas con un radio de 1250 metros. Eso quiere decir que en caso inicial, el usuario en el centro tendrá un margen de notificación de 3750 metros.

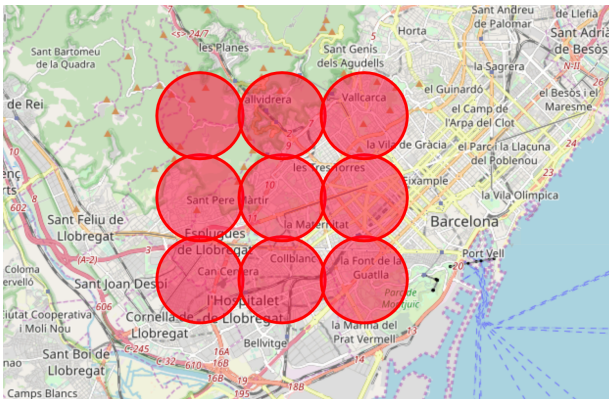


Figura 8.5: Nivel 3 del Grid.

Grid con todos los niveles

A continuación se muestra el Grid con todos los niveles juntos.

Destacar lo pequeño que es el nivel 1 del Grid en comparación con el nivel 3. Por otra parte, la presencia de espacios en blanco en el nivel 3, podría ser un problema para este algoritmo, siempre y cuando no haya una notificación EXIT de cualquier Geofence del nivel 2.

8.4.3. Experimentos efectuados

En esta sección se han definido 3 tipos de experimentos, diferenciados por la velocidad de movimiento del teléfono móvil: andando, corriendo y en coche.

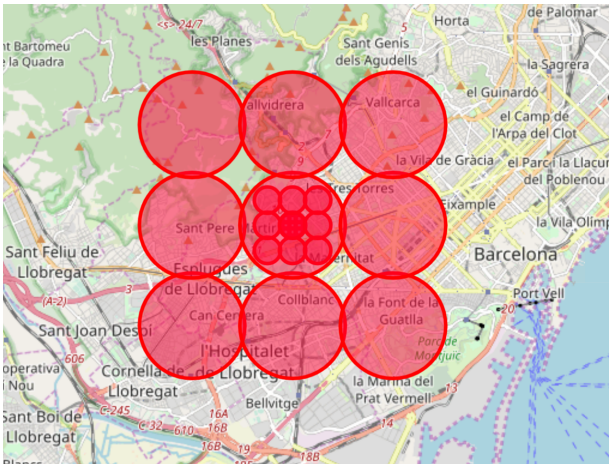


Figura 8.6: Grid con todos los niveles.

Como se ha comentado anteriormente, el margen de maniobra, teniendo al usuario en el centro, es en total de 3750m. Eso quiere decir que, si una notificación se lanza cada 5 minutos, el usuario debe ir a casi 45km/h. Esta velocidad un coche la sobrepasa, pero realmente nuestro algoritmo se volvería a centrar en el caso que se notifique algún EXIT del nivel 1 del Grid

A continuación se muestran los resultados de cada uno de los experimentos efectuados con la aplicación de rastreo. Recordar que, la ejecución de esta aplicación se hace mediante un Push Notification proveniente de Firebase, siempre y cuando el usuario pulse dicha notificación.

Trayecto andando

Solamente viendo el camino del resultado de las diferentes notificaciones, podemos deducir que es el mismo trayecto que los experimentos de las Geofences estáticas.

El resultado de este experimento es muy satisfactorio, ya que es capaz de rastrear el dispositivo móvil perfectamente sin saber el camino que iba a hacer cuando se inició la aplicación. Incluso estos resultados son mejores que algunos de los experimentos del capítulo anterior.

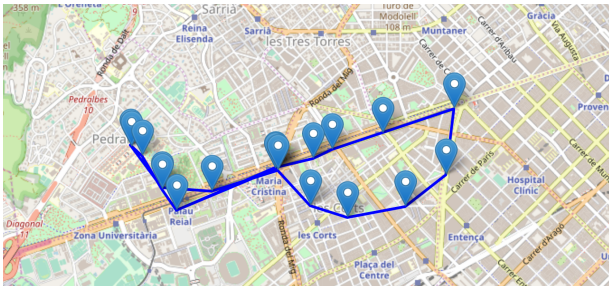


Figura 8.7: Rastreo andando .

Trayecto corriendo

Si medimos la velocidad de una persona corriendo, en media una persona puede ir de 8 a 10 km/h, eso quiere decir que, en 5 minutos puede recorrer hasta 833 metros. En un intervalo de 5 a 10 minutos, puede salir del primer nivel, haciendo que se vuelva a construir todo el Grid. Pero esto solamente ocurriría si la persona fuera siempre en línea recta.

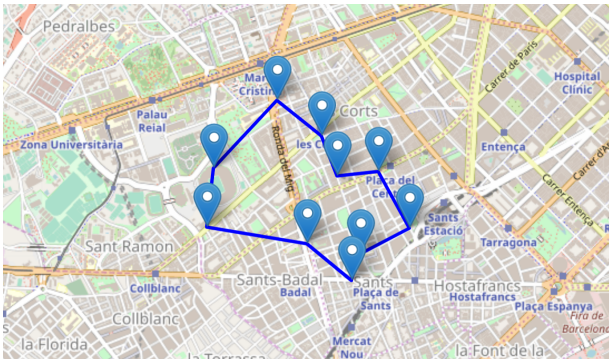


Figura 8.8: Rastreo corriendo.

Durante este recorrido solamente se salió 3 veces del nivel 1, y parece bastante lógico viendo el mapa que tenemos, ya que, el punto de puesta en marcha es el que se encuentra más abajo de la imagen. Además, se puede apreciar la diferencia de 3 zonas de notificación, la inferior, es decir la inicial, el izquierdo, que solamente tenemos dos notificaciones y el derecho que contiene más notificaciones.

Mirando los diferentes registros del sistema, se puede ver que cada una de las 3 zonas es un nuevo Grid situado de nuevo.

Trayecto en coche

El resultado de este experimento es bastante sorprendente, ya que prácticamente se ha conseguido rastrear la trayectoria de un coche a la perfección.

A continuación se muestra el resultado del trayecto en coche:

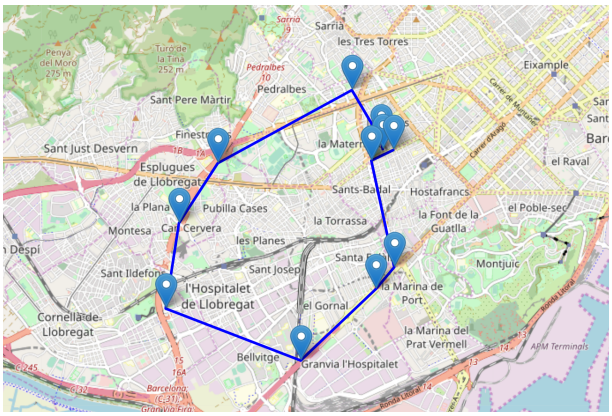


Figura 8.9: Rastreo en coche.

Aunque hay momentos en el que el coche puede llegar hasta a 80km/h en la ronda de Barcelona, se observa que igualmente se han producido transiciones, ya que nunca ha salido del Grid definido. Si el trayecto fuera siempre en línea recta, se podría dudar de su eficacia, ya que el recorrido hecho durante 5 o 10 minutos sería más grande que el radio de todo el Grid.

8.4.4. Push Notification

Como se ha dicho en el apartado de recursos, Push notification se implementa a partir del servicio proporcionado por Firebase, llamado Firebase Cloud Messaging.

A continuación, se muestra el fragmento de código donde se recibe la notificación en caso de que la aplicación esté en primer plano:

```
1  @Override
2      public void onMessageReceived(RemoteMessage remoteMessage) {
3
4          Log.d(TAG, "From: " + remoteMessage.getFrom());
5          // Check if message contains a notification payload.
6          if (remoteMessage.getNotification() != null) {
7              Log.d(TAG, "Message Notification Body: " + remoteMessage.
                   getNotification().getBody());
8          }
```

```
9         // Check if message contains a data payload.
10        if (remoteMessage.getData().size() > 0) {
11            for (String key : remoteMessage.getData().keySet()) {
12                if (key.equals("dynamicGeofence")) {
13                    if (remoteMessage.getData().get(key).equals("run 1")) {
14                        Boolean ok = Constants.set_Geofence_strategy("
                              overlapping");
15                        if (ok) scheduleJob();
16                    }
17                    if (remoteMessage.getData().get(key).equals("run 2")) {
18                        Boolean ok = Constants.set_Geofence_strategy("
                              separated");
19                        if (ok) scheduleJob();
20                    }
21                    if (remoteMessage.getData().get(key).equals("stop")) {
22                        cancelJob();
23                    }
24                }
25            }
26        }
27        sendNotification(remoteMessage.getNotification().getBody());
28    }
```

Cuando la aplicación se está ejecutando en primer plano, la notificación Push no llega a la barra de notificaciones del teléfono móvil, sino que se ejecuta directamente onMessageReceived de la clase FirebaseMessagingService, donde tienes que ser tú quien cree la notificación y la ponga en la barra de notificaciones.

Pocas veces te llega una notificación de una aplicación justo en el momento que se encuentra en primer plano ejecutándose. En la situación que la aplicación estuviera apagada, lo más común sería recibir una notificación en la barra de notificaciones del teléfono móvil, y cuando el usuario pulsara encima se abriría la aplicación ejecutando la función onCreate() de la clase MainActivity. Es en ese momento que se debe mirar si se recibe un Intent y ver qué mensaje contiene. Este caso se muestra en el fragmento de código siguiente.

```
1  if (getIntent().getExtras() != null) {
2      for (String key : getIntent().getExtras().keySet()) {
3          if (key.equals("dynamicGeofence")) {
4              if (getIntent().getExtras().getString(key).equals("run")) {
5                  scheduleJob();
6              }
7              if (getIntent().getExtras().getString(key).equals("stop")) {
8                  try {
9                      cancel();
10                 } catch (IOException e) {
11                     e.printStackTrace();
12                 }
13             }
14         }
15     }
16 }
```

Como se puede observar en los dos extractos anteriores de código, el comportamiento de la aplicación dependerá del contenido del mensaje que lleva la notificación, en forma de clave y valor.

- Si llega la llave 'run', la aplicación crea el servicio en segundo plano el cual construirá el servicio Geofence.
- En el caso que llegue la llave 'stop', cancelará el servicio en segundo plano, destruyendo así el servicio Geofence.

Además, antes de recibir la notificación, se debe añadir el Listener para poderse conectar con Firebase, obteniendo la instancia de FirebaseMessaging. Se presenta a continuación el fragmento de código.

```
FirebaseMessaging.getInstance().subscribeToTopic("dynamicGeofence")
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        String msg = "Succesfull";
                        if (!task.isSuccessful()) {
                            msg = "Failed";
                        }
                        Log.d("Firebase Instance", msg);
                    }
                });
```

Firebase Cloud Messaging es capaz de enviar el mensaje de dos formas: Directamente a un dispositivo físico, sabiendo su token e instancia de la aplicación, o directamente a todos los dispositivos suscritos a un tema.

Como se puede ver en el extracto anterior, la aplicación instalada está suscrita al tema Geofence. Con esta configuración, cada vez que Firebase envíe una notificación, podrá llegar a cualquier teléfono que tenga esta aplicación instalada, por el simple hecho de estar suscrito a Geofence, y como hemos visto anteriormente, consultará qué mensaje contiene para ver qué hacer en cada caso.

8.5. Análisis de los Resultados

En este capítulo, se ha creado un algoritmo capaz de rastrear un dispositivo móvil a partir de pulsar una notificación Push Notification. Además, presenta las siguientes características:

- Rastrear un teléfono móvil Android en cualquier situación, siempre y cuando se encuentre dentro de una ciudad.
- El algoritmo se inicia cuando el usuario deshabilita la restricción de ubicación en segundo plano en el teléfono móvil.

Aparte de todo esto, hemos visto por primera vez en documentación oficial la afirmación de que, una tecnología no funciona con la restricción de segundo plano habilitada, es el caso de la implementación de FCM en Android enviando

mensajes de datos a servicios en segundo plano. Más información en `https://firebase.google.com/docs/cloud-messaging/android/send-multiple?hl=es-419`.

Visto todo esto y finalizando este capítulo, destacamos que se han cumplido todos los objetivos esperados de esta sección, definida como la capacidad de rastrear un dispositivo móvil, y eso quiere decir que, se da por cerrado el Treball de Final de Grau.

Capítulo 9

Conclusiones

9.1. Conclusiones

Como se ha podido ver en detalle en los capítulos anteriores, el resultado del proyecto ha sido muy positivo, alcanzando correctamente todos los objetivos marcados al inicio de esta memoria.

Sin embargo, es necesario señalar que, el objetivo marcado de la obtención de datos de localización en segundo plano, se ha visto afectado por las restricciones tanto de Android como de las interfaces del fabricante. Dicho esto, nuestra implementación software solamente funcionaba cuando el usuario deshabilitaba la restricción de ubicación en segundo plano. Esto es algo que hemos querido controlar desde un principio, pero después de muchas pruebas no hemos podido conseguir una ejecución correcta de ubicación en segundo plano.

Por otra parte, cabe destacar que el inconveniente encontrado no ha hecho que el proyecto siga adelante. Con la ejecución de diferentes experimentos se ha podido estudiar el comportamiento de Geofence delante de la definición de diferentes parámetros y factores. Además, gracias a este estudio, se ha podido diseñar un algoritmo basado en Geofence, capaz de rastrear un teléfono móvil cuando le llega un mensaje de Push Notification.

Finalmente, creo que este proyecto puede ser una herramienta muy útil para el grupo de investigación donde actualmente trabajo, ya que estaban muy interesados en experimentar cómo realmente funcionan las Geofences bajo distintos escenarios, y así poder hacer diversos artículos de investigación reflejando este nuevo sistema tan novedoso como es Geofence.

9.1.1. Relación con el grado

Con el desarrollo de este proyecto, se han intentado cubrir una gran parte de las competencias trabajadas en el grado.

- Se ha diseñado una aplicación software a partir de la elaboración tanto de diagramas UMl como de casos de uso, vistos en la asignatura de Bases de Dades (BD) y a Enginieria del Software (IES).
- Se han utilizado conocimientos adquiridos durante el desarrollo de una aplicación Android en la asignatura de Interacció y Disseny d'Interfícies (IDI).
- Gracias a la experiencia obtenida en Python en la asignatura de Compressió de Dades i Imatges(CDI), se ha podido proceder a un buen tratamiento de datos.
- También se han utilizado conocimientos adquiridos en la asigntura de Internet móvil (IM) para poder caracterizar cada uno de los sistemas de localización de Android y como estos obtienen una ubicación.
- Finalmente, durante la realización de este proyecto, se han trabajado las competencias transversales de uso solvente de los recursos de la información y aprendizaje autónomo. Ya que muchos conocimientos adquiridos han sido fruto de diversos análisis de la información encontrada, con el fin de aprender nuevos métodos y conocimientos para adaptarse a nuevas situaciones.

9.1.2. Aspectos a contemplar en una segunda fase del proyecto

A continuación se presentan aspectos que se deberían contemplar en una segunda fase del proyecto:

- Disponer de más parámetros para analizar otros aspectos en el comportamiento de Geofence. Por ejemplo analizar las diferentes prioridades del proveedor de ubicación o probar con diferentes valores en los intervalos de obtención de ubicación. Esto no ha sido posible ya que el número de experimentos se disiparía quedando fuera de nuestro alcance.
- En el análisis de los datos de Geofence se debería dar un enfoque más en profundidad en todos los datos obtenidos, ya que se podrían encontrar otros estados que a simple vista no se apreciaban.
- Analizar más profundamente el tiempo de media en la obtención de ubicación en segundo plano en Geofence, puesto que siempre tiene un intervalo medio de 5 minutos. Algo que contradice Google, ya que te dice que en segundo plano Geofence es capaz de obtener localización entre 2 o 3 minutos.

9.2. Opinión Personal

Personalmente, estoy muy contento con el resultado de este proyecto. Con su realización me ha permitido poner en práctica numerosos conocimientos adquiridos durante la realización del grado, y además, se han podido profundizar temas vistos de manera muy superficial, como es el caso de la programación en Android.

Por otra parte, este trabajo me ha permitido conocer herramientas muy interesantes para llevar a cabo un proyecto de características similares, y que probablemente podrían ser muy útiles durante mi carrera profesional.

En último lugar, la realización de este proyecto me ha hecho adquirir aspectos a nivel personal muy positivos, tanto en conocimientos como en forma de encarar diversas situaciones. Además, quiero destacar que puedo estar muy orgulloso del desenlace de este proyecto.

9.3. Propuestas para un Futuro

A continuación, se detalla alguna propuesta de trabajos futuros a realizar a partir de este proyecto:

- Por el motivo de que no existen técnicas de rastreo basadas en Geofence, se cree que durante el desarrollo del algoritmo de rastreo se debería haber hecho alguna versión más para poder comparar diferentes resultados. Por tema de planificación, se ha decidido poner como propuesta para un trabajo futuro.
- Desarrollo de un algoritmo adaptativo capaz de encontrar puntos de interés de las personas. Se ha pensado en un algoritmo que vaya disminuyendo el tamaño de las Geofences hasta encontrar puntos dónde una persona pasa la mayoría del día. Por ejemplo, se podría saber la localización del lugar en el que vive, lugar trabajo, etc.

Bibliografía

[1] Samuel Fernández. (Agosto 10, 2018). Así era el mercado móvil en 2008. https://www.xatakamovil.com/mercado/asi-era-mercado-movil-2008-ano-su-nacimiento-llegada-android-9-pie

[2] Big Data: ¿En qué consiste? (Febrero 25,2019). https://www.powerdata.es/big-data

[3] Pardo, Dimas (Febrero 14, 2019). ¿Para qué sirve una API? Solventa por fin esta duda de primerizo. https://blog.pandorafms.org/es/para-que-sirve-una-api/

[4] Desarrollo de Aplicaciones para Android (Febrero 25,219). http://www.jtech.ua.es/cursos/apuntes/moviles/daa2013/wholesite.pdf

[5] Google APIs for Android. Fused Location. (Marzo 21, 2018). https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient

[6] Foreground-Background. (Febrero 25, 2019). https://www.pcmag.com/encyclopedia/term/43390/foreground-background

[7] Límites de ubicación en segundo plano. (Febrero 25, 2019). https://developer.android.com/about/versions/oreo/background-location-limits?hl=es-419

[8] Optimize location for battery. (19 Febrero, 2019). https://developer.android.com/guide/topics/location/battery

[9] Google Developers India. (Diciembre 1, 2017).Location and Battery (GDD India '17). https://www.youtube.com/watch?v=_HaL7jdQTVg

[10] Natalia Wawrzyniak and Tomasz Hyla. (Agosto 23, 2016). Application of Geofencing Technology for the Purpose of Spatial Analyses in Inland Mobile Navigation. https://ieeexplore.ieee.org/document/7548001/metrics#metrics

[11] Flavio Cirillo, Tobias Jacobs, Miquel Martin, and Piotr Szczytowski. (Septiembre 25, 2014). Large Scale Indexing of Geofences. https://ieeexplore.ieee.org/document/6910110

[12] GDPR: ¿qué necesitas saber del nuevo Reglamento Europeo de Protección de Datos? (Enero 4, 2018). https://blog.signaturit.com/es/las-claves-sobre-el-nuevo-reglamento-europeo-de-proteccion-de-datos

[13] Guía de Protección de Datos para desarrolladores de aplicaciones móviles (Mayo 29, 2019). https://ayudaleyprotecciondatos.es/2016/06/06/normativa-lopd-aplicaciones-moviles/

[14] Optimize location for battery. (Febrero 25, 2019). https://developers.google.com/android/reference/com/google/android/gms/location/Geofence.Builder.

[15] Flavio Cirillo, Tobias Jacobs, Miquel Martin, and Piotr Szczytowski. (Agosto 4, 2014). Large Scale Indexing of Geofences. https://ieeexplore.ieee.org/document/6910110

[16] Spyzie. (Febrero 25, 2018). What can I do with the geofencing features of Spyzie?. https://www.spyzie.com/monitoring-geofences.html.

[17] Axel Küpper, Ulrich Bareth, and Behrend Freese. (Octubre 7, 2011).Geofencing and Background Tracking – The Next Features in LBSs. https://www.user.tu-berlin.de/komm/CD/paper/010221.pdf.

[18] Create and monitor geofences. (Febrero 25, 2019). https://developer.android.com/guide/topics/location/battery

[19] Precio kWh España (Marzo 6, 2019). https://preciogas.com/faq/precio-kwh-espana

[20] Pràctiques en empresa (Marzo 8, 2019). https://www.fib.upc.edu/ca/empresa/practiques-en-empresa

[21] OLD EN: Indirect costs in Horizon 2020a (Marzo 8, 2019). https://www.ffg.at/en/europe/legalandfinancialmatters/h2020_indirect-costs

[22] What is Horizon 2020? (Marzo 8, 2019). https://ec.europa.eu/programmes/horizon2020/what-horizon-2020

[23] Latitude and Longitude. (Mayo 30, 2019). http://www.ketteringschools.org/userfiles/1375/Classes/13956/LatitudeandLongitude.pdf

[24] GPS Accuracy. (Mayo 30,2019) https://www.gps.gov/systems/gps/performance/accuracy/how-accurate

[25] WiFi 802.11mc: ¿cómo funciona el GPS para interiores?. (Mayo 30, 2019) https://www.adslzone.net/2018/03/21/802-11mc-gps-interior/

[26] Límites de ubicación en segundo plano, Android Developers. (Mayo 31, 2019) https://developer.android.com/about/versions/oreo/background-location-limits?hl=es-419

[27] ¿Qué es una API y para qué sirve? (Febrero 16, 2015) https://www.abc.es/tecnologia/consultorio/20150216/abci–201502132105.html

[28] How to Check Sensors and Hardware on Android and iPhone. (Mayo 31, 2019). https://howtotechnaija.com/check-sensors-and-hardware-android-iphone/

[29] Simple, battery-efficient location API for Android. (Mayo 31, 2019). https://developers.google.com/location-context/fused-location-provider/

[30] Provide contextual experiences when users enter or leave an area of interest. (Mayo 31, 2019). https://developers.google.com/location-context/geofencing/

[31] GPS: location-tracking technology. (Abril, 2002) Publicado en: Computer (Volumen: 35)

[32] ¿Cómo funcionan los dispositivos GPS? Trilateración vs Triangulación. (Mayo 31, 2019). https://acolita.com/como-funcionan-los-dispositivos-gps-trilateracion-vs-triangulacion/

[33] WiFi-based location services attack on dual-band hardware. (Agosto 8, 2014). Jun Liang Roy Feng y Guang Gong

[34] Wi-Fi Location: Qué es, cómo funciona y para qué sirve este estándar de geoposicionamiento en interiores con Wi-Fi. Escrito por Sergio De Luz (Marzo 11, 2017) https://www.redeszone.net/2017/03/11/wi-fi-location-funciona-sirve-este-estandar-geoposicionamiento-interiores-wi-fi/

[35] Introducción a WorkManager. (Enero, 18, 2019) https://developers-latam.googleblog.com/2019/01/introduccion-workmanager.html

[36] Background Check and Other Insights into the Android Operating System Framework (Google I/O '17), (Mayo 17, 2017) https://www.youtube.com/watch?v=hbLAzwhBjFEt=663s

[37] Improving Battery Life with Restrictions (Android Dev Summit '18), (Noviembre 18, 2018) https://www.youtube.com/watch?v=hbLAzwhBjFEt=663s

[38] Límites de ubicación en segundo plano. (Mayo 31, 2019). https://developer.android.com/about/versions/oreo/background-location-limits?hl=es-419

[39] Novedades de Android Studio 2.0. (Mayo 31, 2019). https://www.armadilloamarillo.com/blog/novedades-de-android-studio-2-0/

[40] https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse. (Mayo 31, 2019). https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/

[41] Límites de ubicación en segundo plano. (Mayo 06, 2019) https://developer.android.com/about/versions/oreo/background-location-limits.html

[42] Android Push Notifications using Firebase Cloud Messaging (FCM). (Octubre 23, 2016). https://www.coderefer.com/android-push-notifications-fcm/

[43] Neha Srivastava, Uma Shree, Nupa Ram Chauhan, Dinesh Kumar Tiwari. (Mayo 9, 2017). FIREBASE CLOUDMESSAGING(ANDROID) http://www.ijirset.com/upload/2017/cotii/3_CS_COTII_2017_Firebase_cloud.pdf

[44] Envía mensajes a varios dispositivos. (Mayo 13, 2019). https://firebase.google.com/docs/cloud-messaging/android/send-multiple?hl=es-419

[45] Adnroid Background Services. (Mayo 13, 2019). http://mobdev.ce.unipr.it/2013/download/[9]

[46] De los datos al Big Data (Mayo 13, 2019). https://www.evaluandosoftware.com/de-los-datos-al-big-data/

Índice de figuras

Índice de cuadros